CSC 480/580 Principles of Machine Learning

05 Practical Considerations

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA

Outline

- The role of features in supervised learning
- Classification metrics beyond error rate
- Model Evaluation & Comparison
- Debugging Learning Algorithms
- Bias / Variance Tradeoff

The role of features in supervised learning

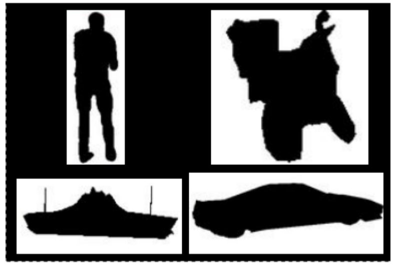
The role of features in supervised learning

- **Importance of feature representation**
- Feature selection
- Feature transformation

The importance of good feature representation

- Pixel representation:
  - represent an image as a w*h*3 dimensional vector
  - treat all coordinates in the same role
  - throw away all locality information in the image

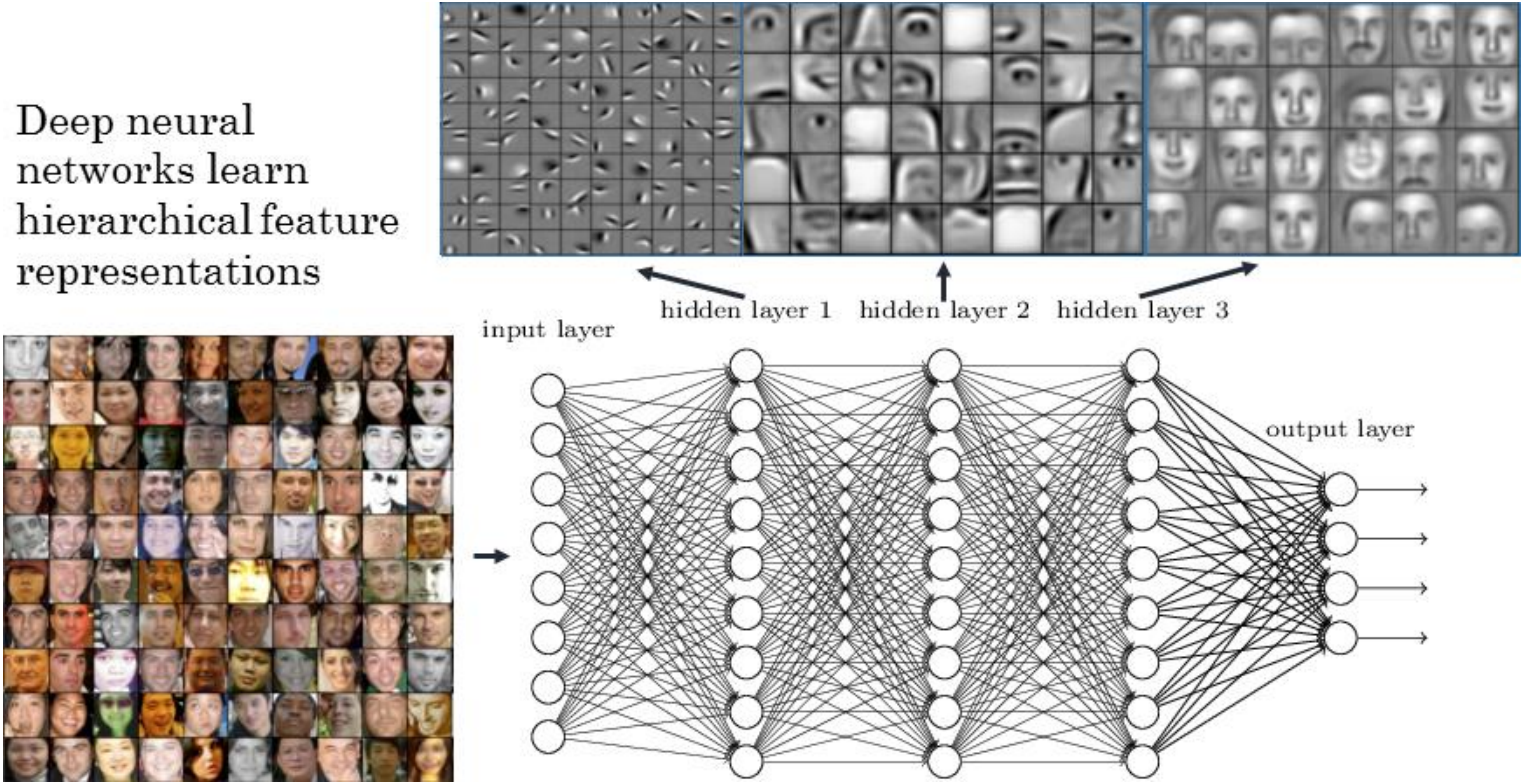
- Shape representation:
  - represent a colored image with a w*h black-white image



- Bag-of-words representation:

	free	offer	lecture	cs	Spam?
Email 1	2	1	0	0	+1
Email 2	0	1	3	1	-1

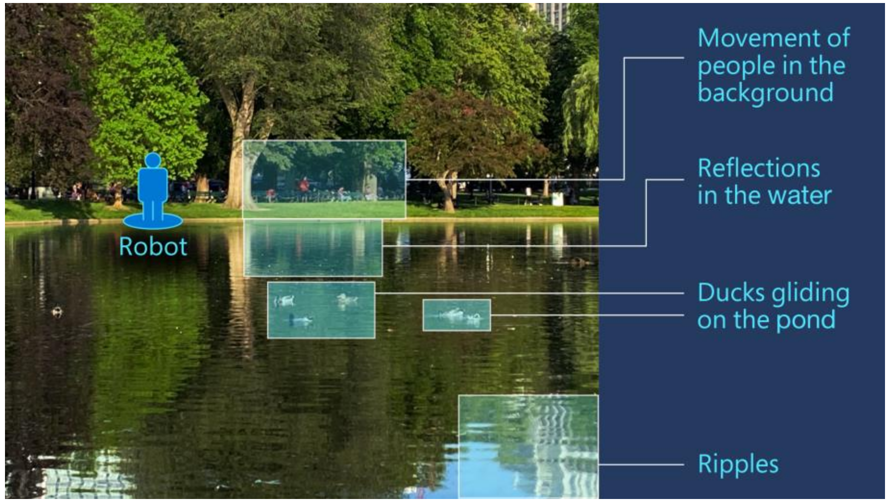
Deep neural networks learn hierarchical feature representations



https://www.rsipvision.com/exploring-deep-learning/

Irrelevant and redundant features

- Irrelevant features
  - y is independent of f
  - y = Road walkability, f = duck activities in the pond


Movement of people in the background
Reflections in the water
Robot
Ducks gliding on the pond
Ripples

  - If #features is large and #examples is small ⇒ spurious correlation between some feature & label

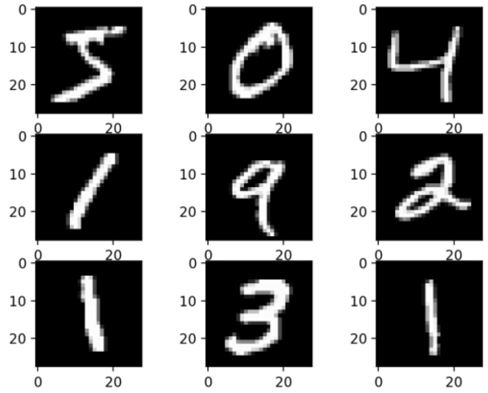
- Redundant features
  - Given f_1, y is (nearly) independent of f_2
  - E.g. detecting fire hydrants

- Learning decision trees implicitly handles these two issues
- How about nearest neighbors / Perceptron?

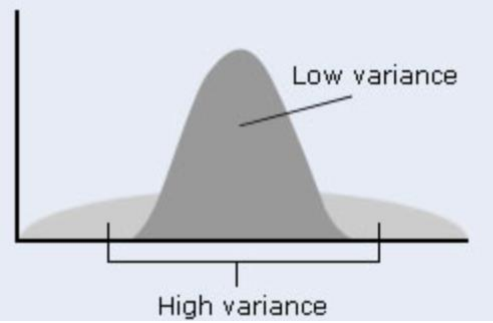
https://www.microsoft.com/en-us/research/blog/ppe-a-fast-and-provably-efficient-rl-algorithm-for-exogenous-noise/

Feature pruning

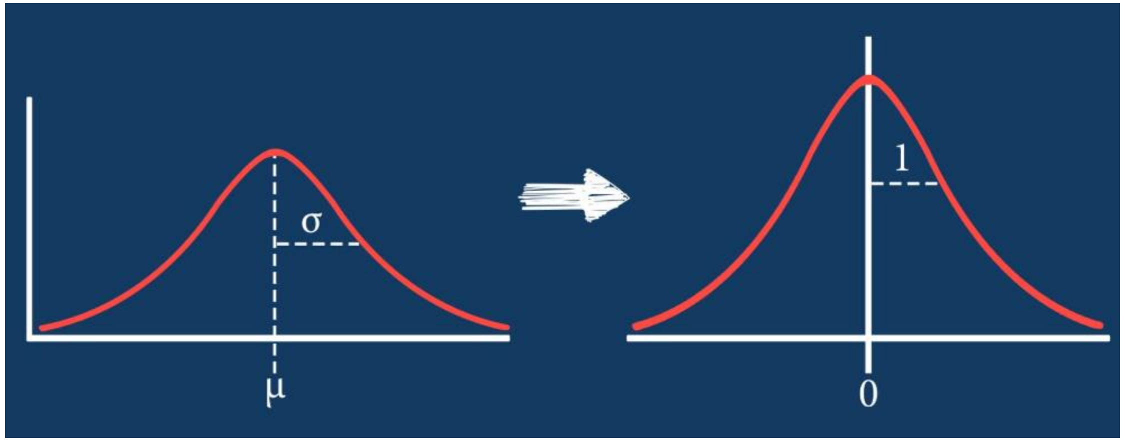
- Removing features that are not very useful for prediction
  - E.g. text classification with bag-of-word representation, remove words that appear in <= K docs
  - E.g. digit classification, remove pixels with low variance



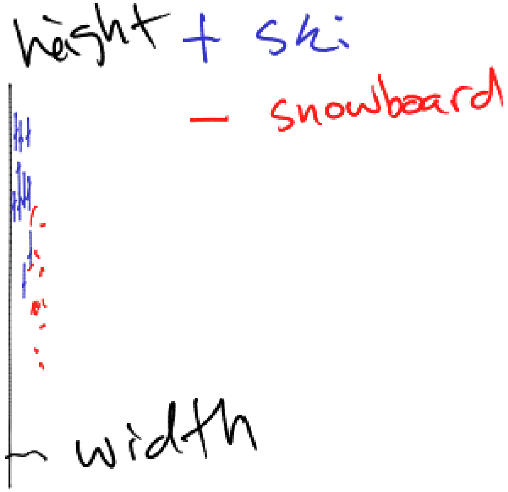
$$\mu_f = \frac{1}{N}\sum_{i=1}^{N} x_{i,f} \quad \sigma_f^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,f} - \mu_f)^2$$



Feature normalization



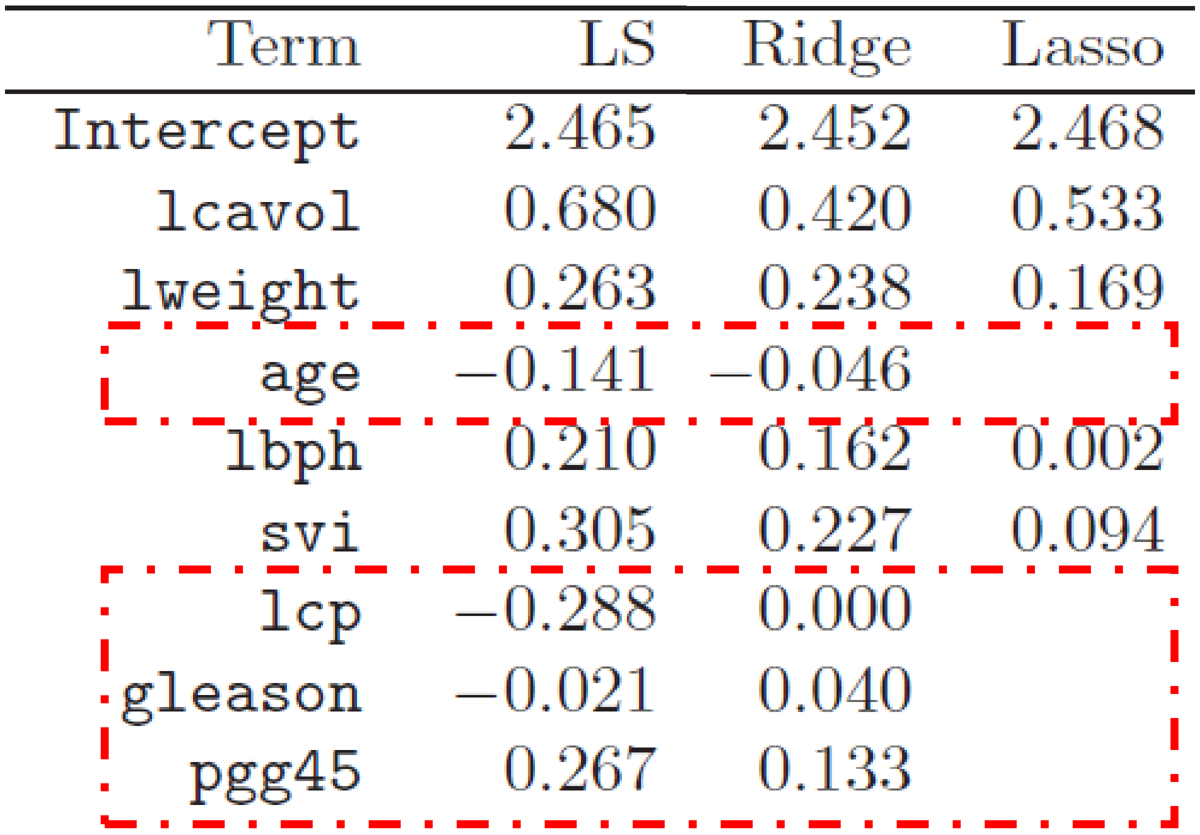
- Centering:
  - $x'_{i,f} = x_{i,f} - \mu_f \;\; \Rightarrow \mu'_f = 0$
- Variance scaling:
  - $x'_{i,f} = x_{i,f}/\sigma_f \;\Rightarrow (\sigma'_f)^2 = 1$
- Absolute scaling
  - $x'_{i,f} = x_{i,f}/r_f$, where $r_f = \max_i |x_{i,f}| \;\Rightarrow$ range of $x'_{i,f}$ in [-1,+1]
- Same transformation applied to both training set and test data
- Aside: example normalization: $x'_i = \frac{x_i}{\|x_i\|}$ sometimes also can be applied



The role of features in supervised learning

- Importance of feature representation
- **Feature selection**
- Feature transformation

Example: Prostate Cancer Dataset



Term	LS	Ridge	Lasso
Intercept	2.465	2.452	2.468
lcavol	0.680	0.420	0.533
lweight	0.263	0.238	0.169
age	−0.141	−0.046	
lbph	0.210	0.162	0.002
svi	0.305	0.227	0.094
lcp	−0.288	0.000	
gleason	−0.021	0.040	
pgg45	0.267	0.133	

Best LASSO model learns to ignore several features (age, lcp, gleason, pgg45).

Wait…Is **age** really not a significant predictor of prostate cancer? What's going on here?

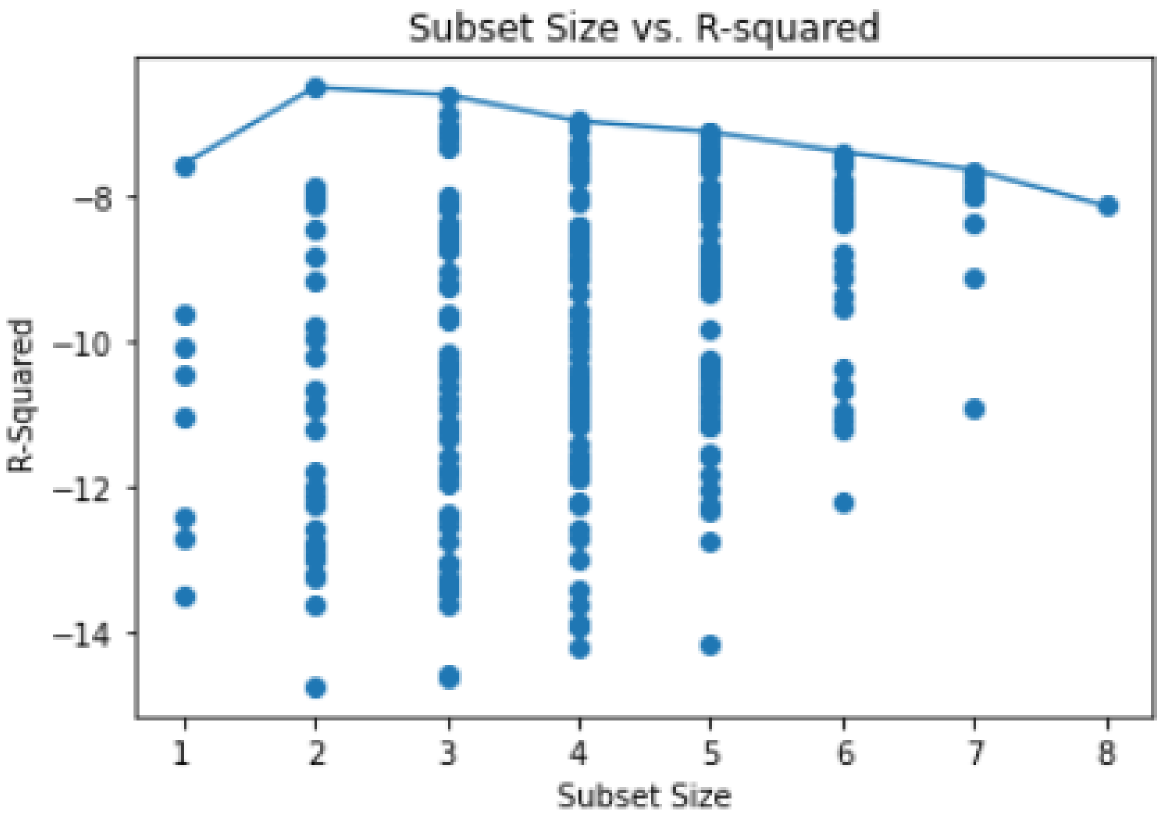
Age is highly correlated with other factors and thus *not significant* in the presence of those factors

Best-Subset Feature Selection

The optimal strategy for p features looks at models over *all possible combinations* of features,

```
For k in 1,…,p:
  subset = Compute all subset of k-features (p-choose-k)
  For kfeat in subset:
    model = Train model on kfeat features
    score = Evaluate model using cross-validation
Choose the model with best cross-validation score
```

Best-Subset Feature Selection : Prostate Cancer Dataset



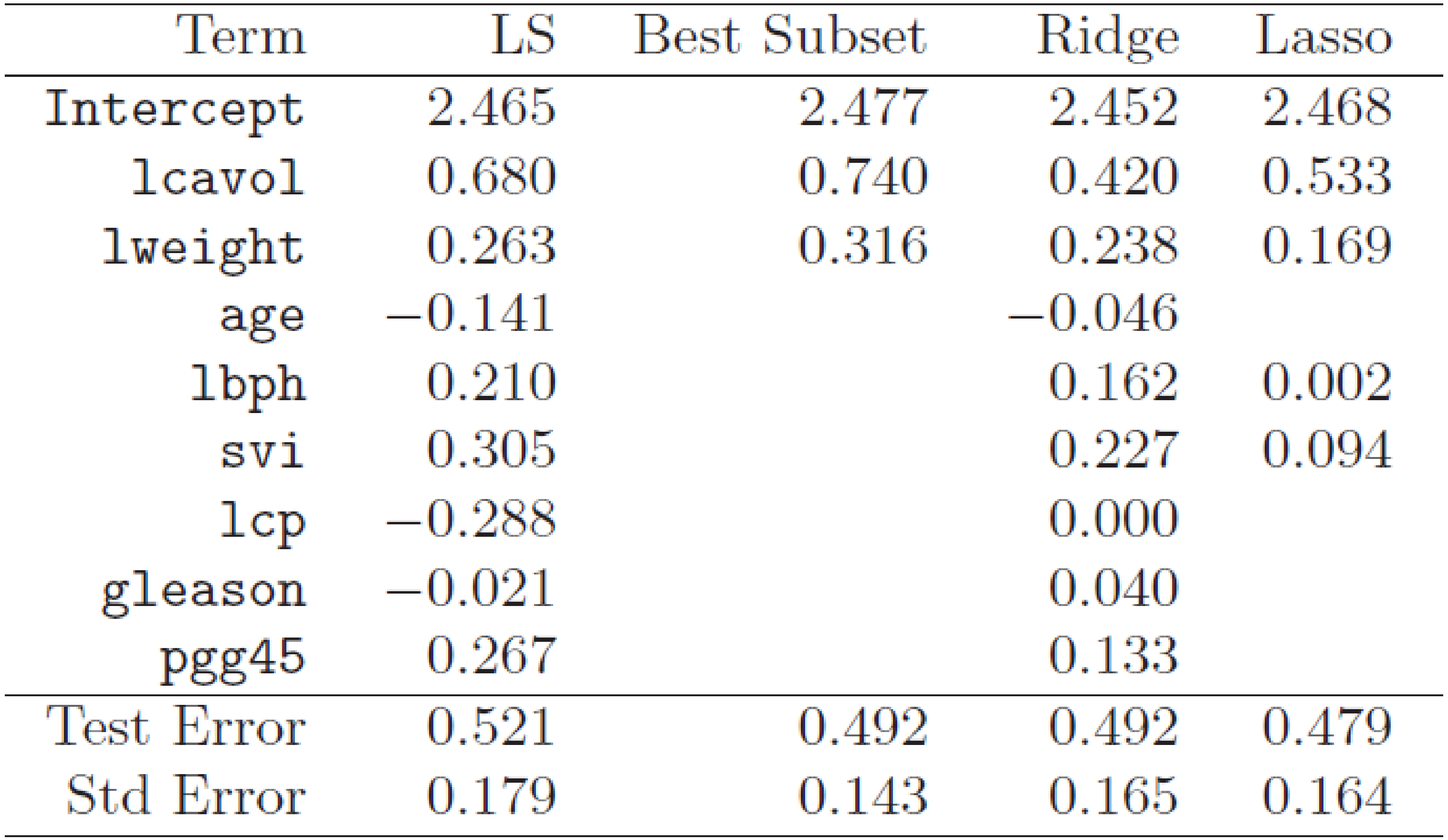
Each marker is the cross-val R^2 score of a trained model for a subset of features

Data have 8 features, there are 8-choose-k subsets for each k=1,…,8 for a total of 255 models

Using 10-fold cross-val requires 10 x 255 = 2,550 training runs!

Feature Selection: Prostate Cancer Dataset

Best subset has lowest test error (lowest variance) with just 2 features



Term	LS	Best Subset	Ridge	Lasso
Intercept	2.465	2.477	2.452	2.468
lcavol	0.680	0.740	0.420	0.533
lweight	0.263	0.316	0.238	0.169
age	−0.141		−0.046	
lbph	0.210		0.162	0.002
svi	0.305		0.227	0.094
lcp	−0.288		0.000	
gleason	−0.021		0.040	
pgg45	0.267		0.133	
Test Error	0.521	0.492	0.492	0.479
Std Error	0.179	0.143	0.165	0.164

[Source: Hastie et al. (2001)]

Forward Sequential Selection

An efficient method adds the most predictive feature one-by-one

```
featSel = empty
featUnsel = All features
For iter in 1,…,p:
  For kfeat in featUnsel:
    thisFeat = featSel + kfeat
    model = Train model on thisFeat features
    score = Evaluate model using cross-validation
  featSel = featSel + best scoring feature
  featUnsel = featUnsel − best scoring feature
Choose the model with best cross-validation score
```

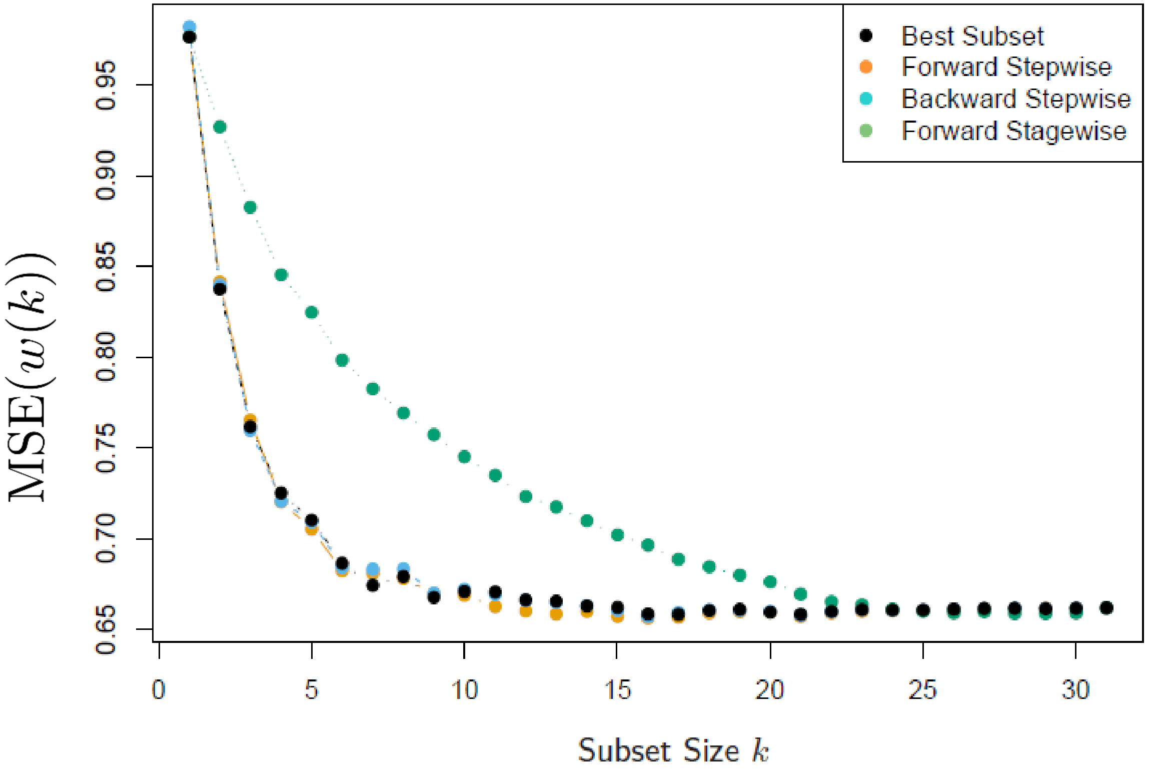
Backward Sequential Selection

Backwards approach starts with *all* features and removes one-by-one

```
featSel = All features
For iter in 1,…,p:
  For kfeat in featSel:
    thisFeat = featSel - kfeat
    model = Train model on thisFeat features
    score = Evaluate model using cross-validation
  featSel = featSel – worst scoring feature
Choose the model with best cross-validation score
```

Comparing Feature Selection Methods

Sequential selection is greedy, but often performs well…



Example Feature selection on synthetic model with p=30 features with pairwise correlations (0.85). True feature weights are all zero except for 10 features, with weights drawn from N(0,6.25).

Sequential selection with p features takes O(p^2) time, compared to exponential time for best subset

Sequential feature selection available in Scikit-Learn under:

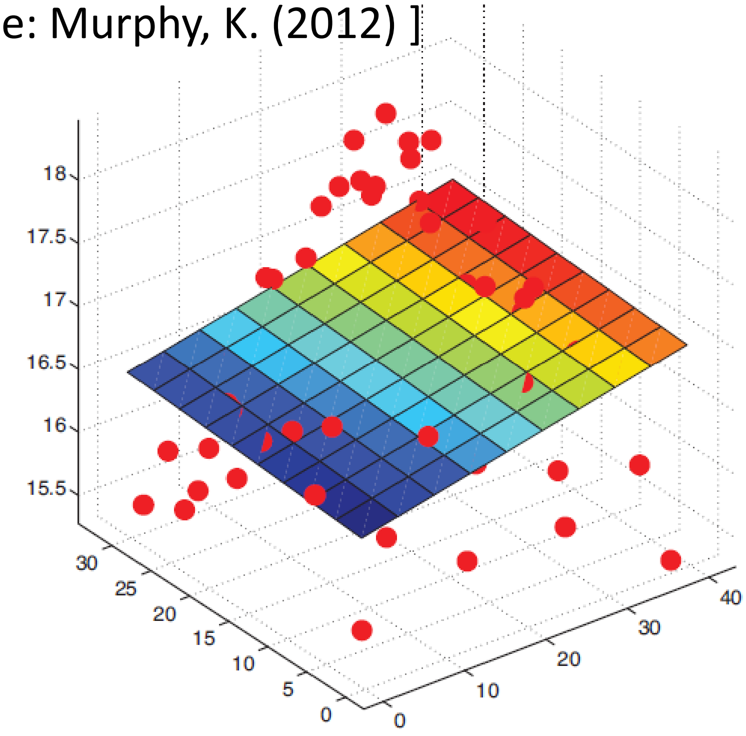
`feature_selection.SequentialFeatureSelector`

The role of features in supervised learning

- Importance of feature representation
- Feature selection
- **Feature transformation**

Linear Models

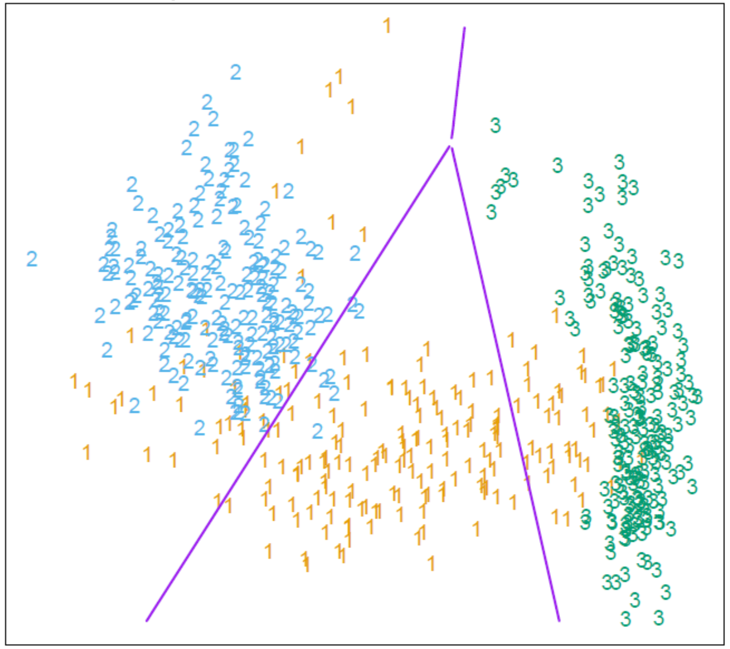
[Image: Murphy, K. (2012)]



Linear Regression Fit a *linear function* to the data,

$$y = w^T x + b$$

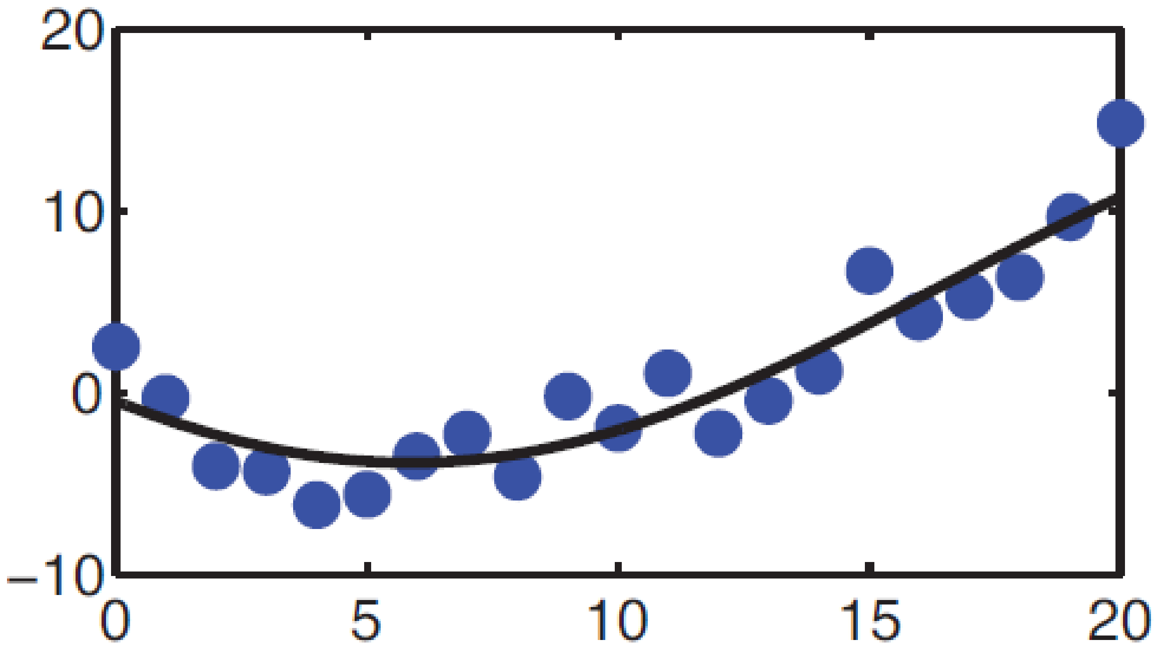
[Image: Hastie et al. (2001)]



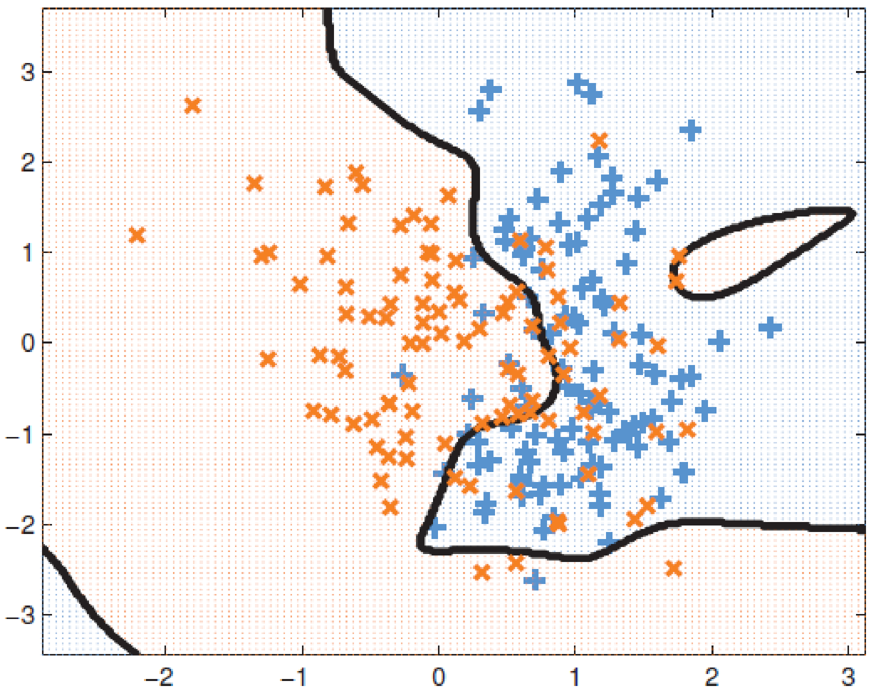
Linear Classification Learn a decision boundary that is *linear in the data*

$$y = \text{sign}(w^T x + b)$$

Nonlinear Data



What if our data are *not* well-described by a linear function?



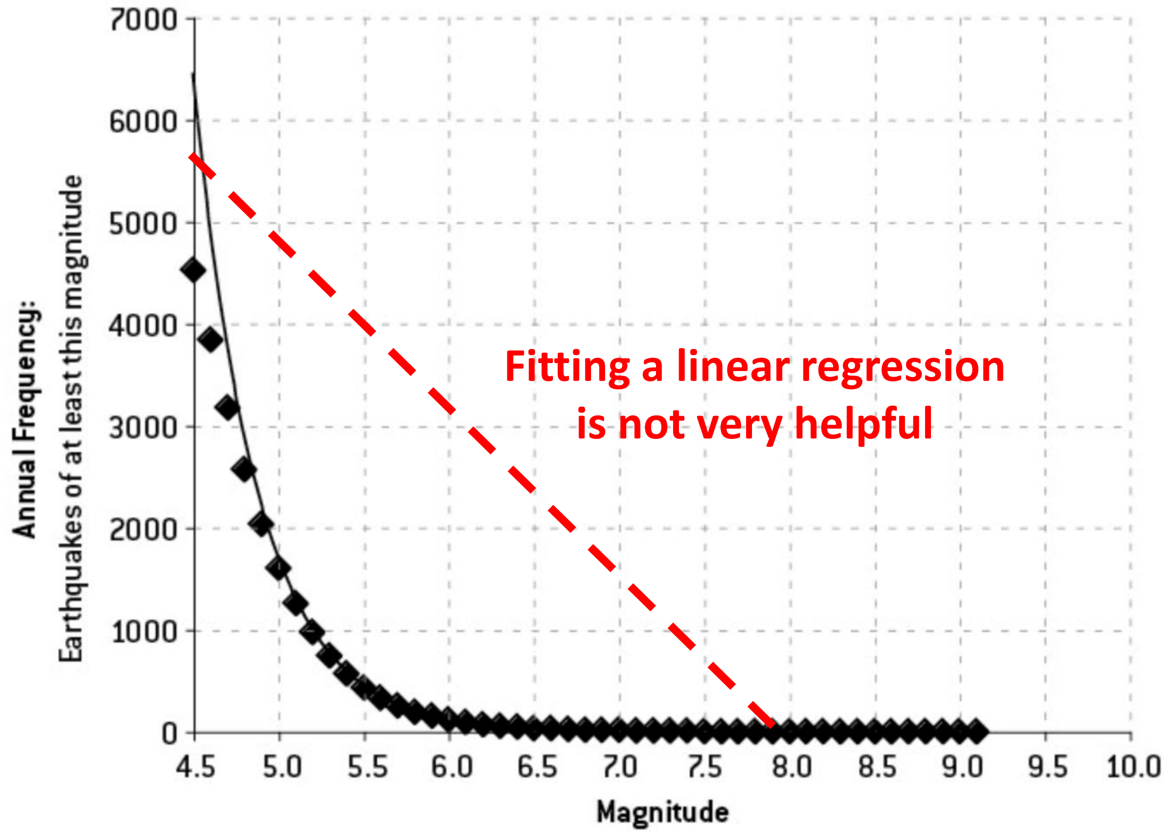
What if classes are *not linearly-separable*?

[Source: Murphy, K. (2012)]

Example: Earthquake Prediction

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3A: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012



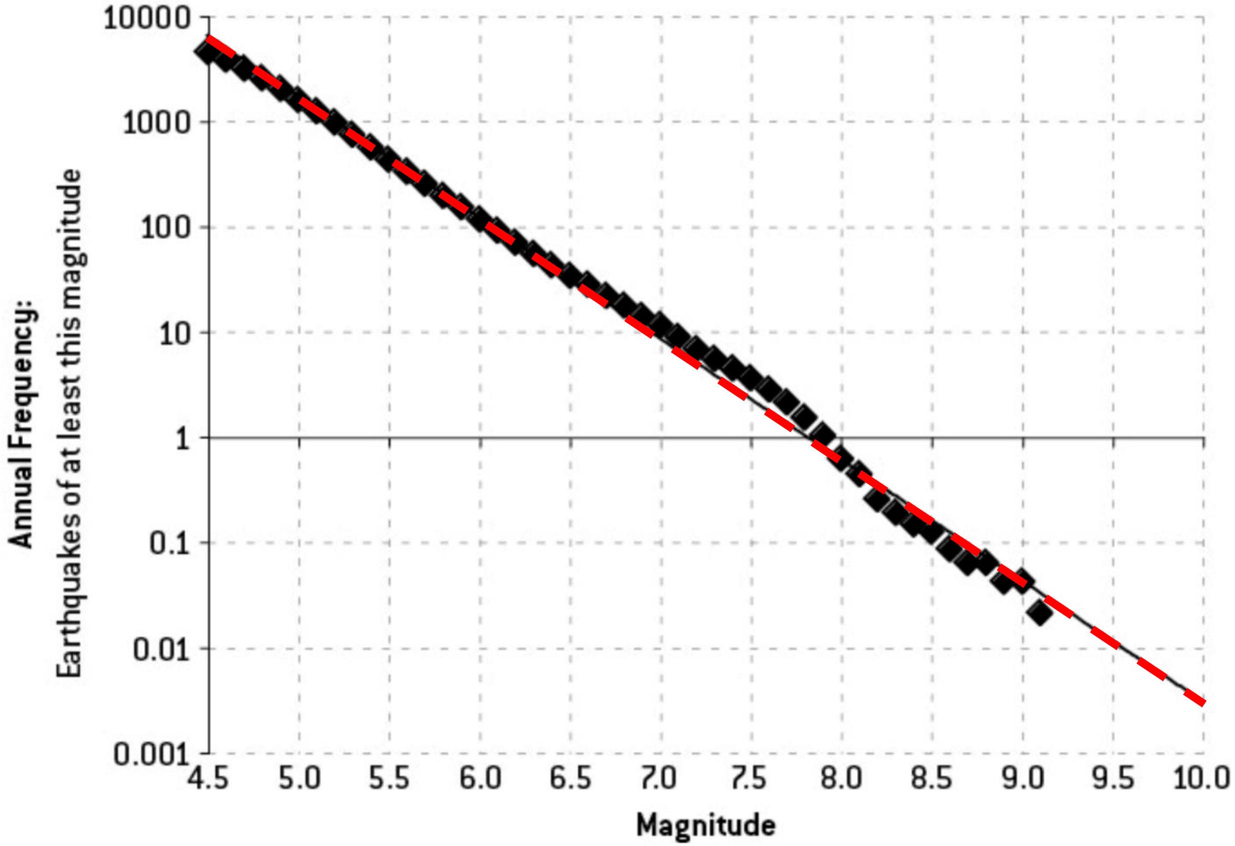
Fitting a linear regression is not very helpful

[Source: Silver, N. (2012)]

Example: Earthquake Prediction

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3B: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012, LOGARITHMIC SCALE



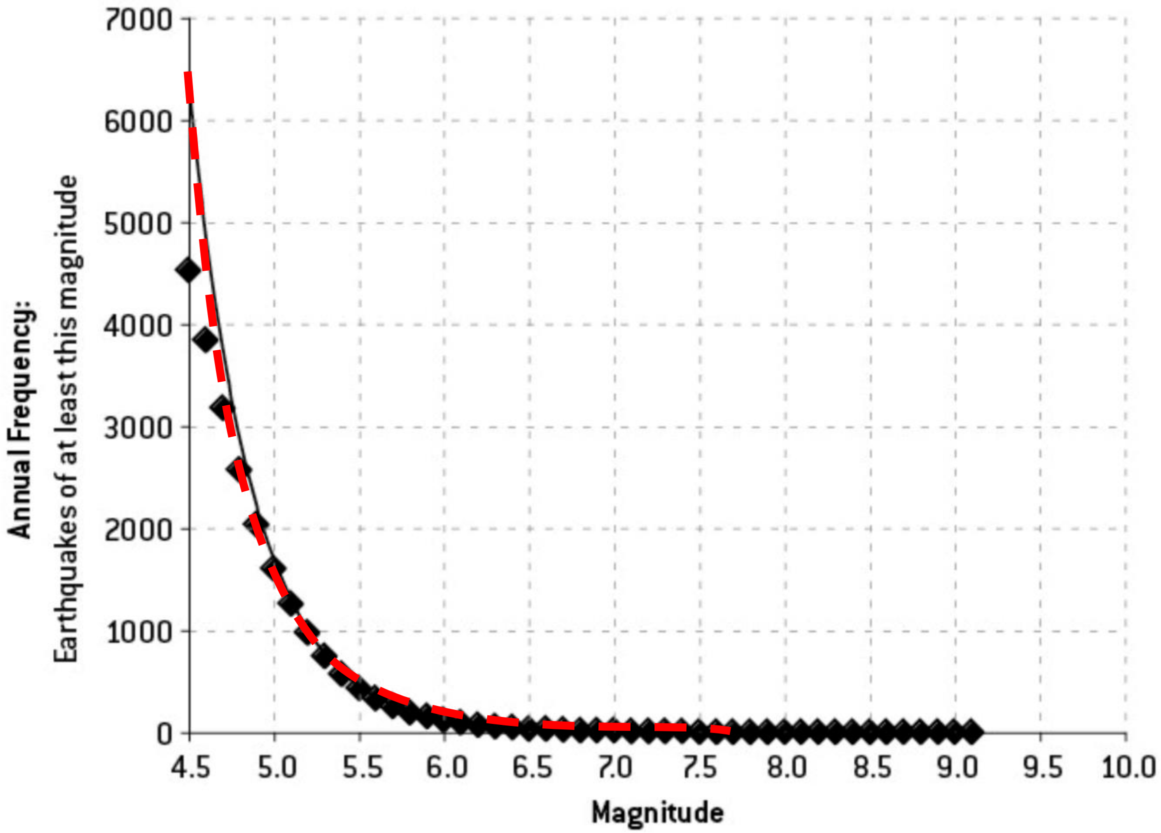
But plotting outputs on a logarithmic scale reveals a strong linear relationship…

[Source: Silver, N. (2012)]

Example: Earthquake Prediction

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3A: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012



Idea Instead of fitting ordinary linear regression,

$$y = w^T x$$

First take the logarithm of output values y,

$$y = w^T \log(x)$$

Or take the exponential of input values x,

$$y = w^T \exp(x)$$

[Source: Silver, N. (2012)]

Basis Functions

- A **basis function** can be any function of the input features X
- Define a set of m basis functions $\phi_1(x), \ldots, \phi_m(x)$
- Fit a linear regression model in terms of basis functions,

$$y = \sum_{i=1}^{m} w_i \phi_i(x) = w^T \phi(x)$$

- Regression model is *linear in the basis transformations*
- Model is *nonlinear in the data X*

Common "All-Purpose" Basis Functions

- Linear basis functions recover the original linear model,

$$\phi_m(x) = x_m$$

Returns m^th dimension of X

- Quadratic. $\phi_m(x) = x_j^2$ or $\phi_m(x) = x_j x_k$ capture 2nd order interactions
- An order p polynomial $\phi \rightarrow x_d, x_d^2, \ldots, x_d^p$ captures higher-order nonlinearities (but requires $O(d^p)$ parameters)
- Nonlinear transformation of single inputs,

$$\phi \rightarrow (\log(x_j), \sqrt{x_j}, \ldots)$$

- An indicator function specifies a region of the input,

$$\phi_m(x) = I(L_m \leq x_k < U_m)$$

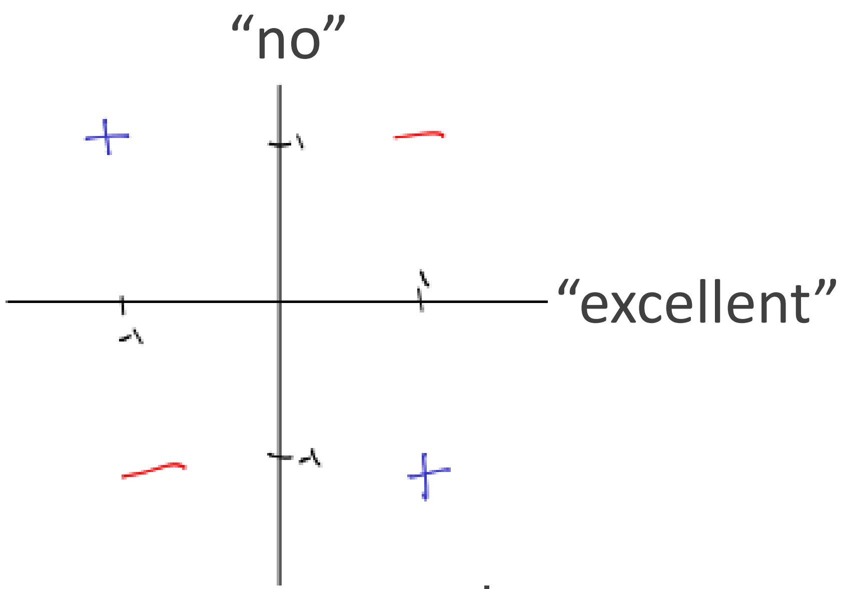
Feature transformations

- Combining features into a “meta-feature”, e.g. $x_{\text{no}} \cdot x_{\text{excellent}}$
  - Useful for e.g. Perceptron learners

- In general, $\binom{d}{k}$ meta-features if allowed to combine k features

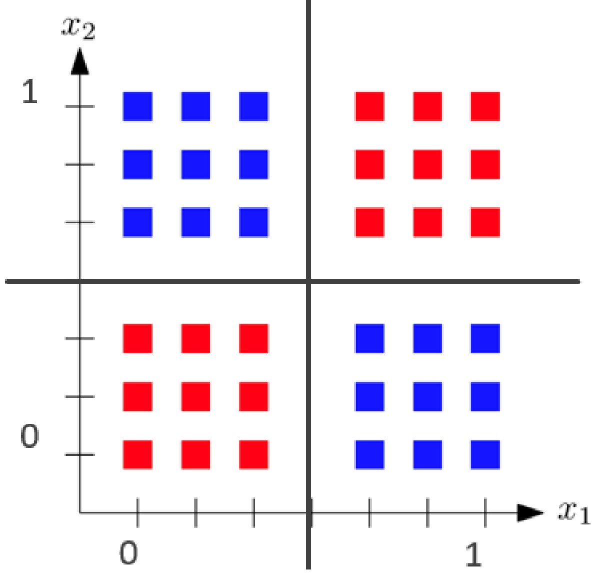
- Computationally cheaper alternative:
  - train a decision tree, use the meta-feature induced by leaves

- Logarithmic feature transformation
  - $x'_f \leftarrow \log_2(x_f)$ (“excellent” word count: 1->2 vs. 10->11)
  - $x'_f \leftarrow \log_2(x_f + 1)$



“no”

“excellent”



CSC 480/580 Principles of Machine Learning

05 Practical Considerations

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA

Outline

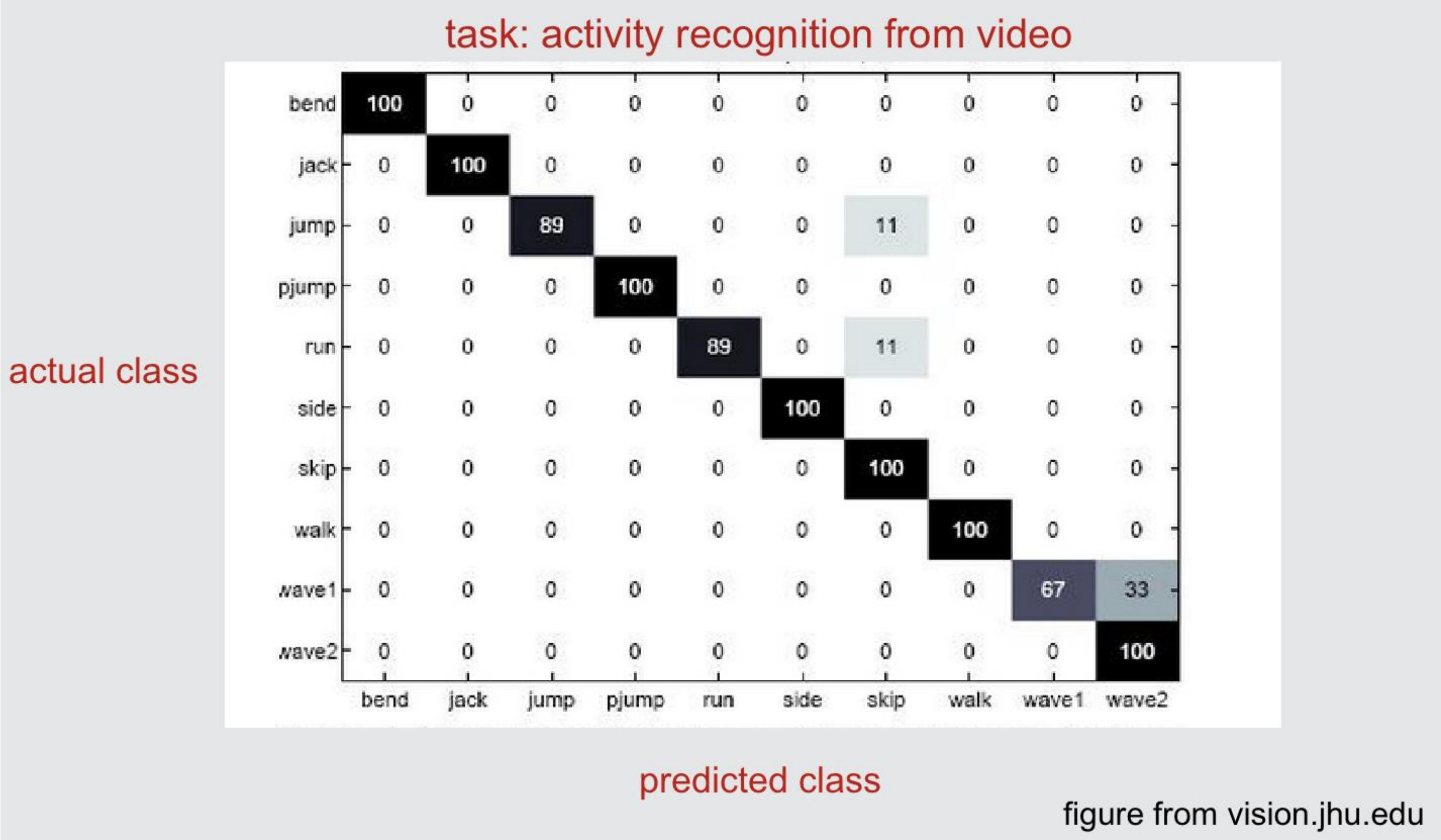
- Classification metrics beyond error rate
- Model Evaluation & Comparison
- Debugging Learning Algorithms
- Bias / Variance Tradeoff

Classification metrics beyond error rate

Confusion matrix

- E.g. activity recognition

- $P(\,\hat{y} = \text{skip} \mid y = \text{jump}\,) = 11\%$



task: activity recognition from video

actual class

predicted class

	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

figure from vision.jhu.edu

Class imbalance problem

- E.g., 5% pos, 95% negative.
- Implicit assumption:

  misclassifying positive example is more costly than misclassifying negative examples

- Standard ML algorithm aims to find h that minimizes unweighted training error $\sum_{i=1}^{n} I(h(x_i) \neq y_i)$
  - Tend to produce classifiers that always predict negative

- 2 alternatives:
  - Duplicate the minority class to make the positive and negative class balanced

    repeat every positive example w times, where $w = P(y = -1)/P(y = +1)$
  - Importance weighted classification: minimize $\sum_{i=1}^{n} w_i \, I(h(x_i) \neq y_i)$,

    where $w_i = 1$ when $y_i = -1$, $w_i = w$ when $y_i = +1$

New measures of classification performance

- True positive rate (TPR)

  $= \frac{\mathrm{TP}}{\mathrm{P}} = \frac{P(\hat{y}=+1, y=+1)}{P(y=+1)}$

  (aka recall, sensitivity)

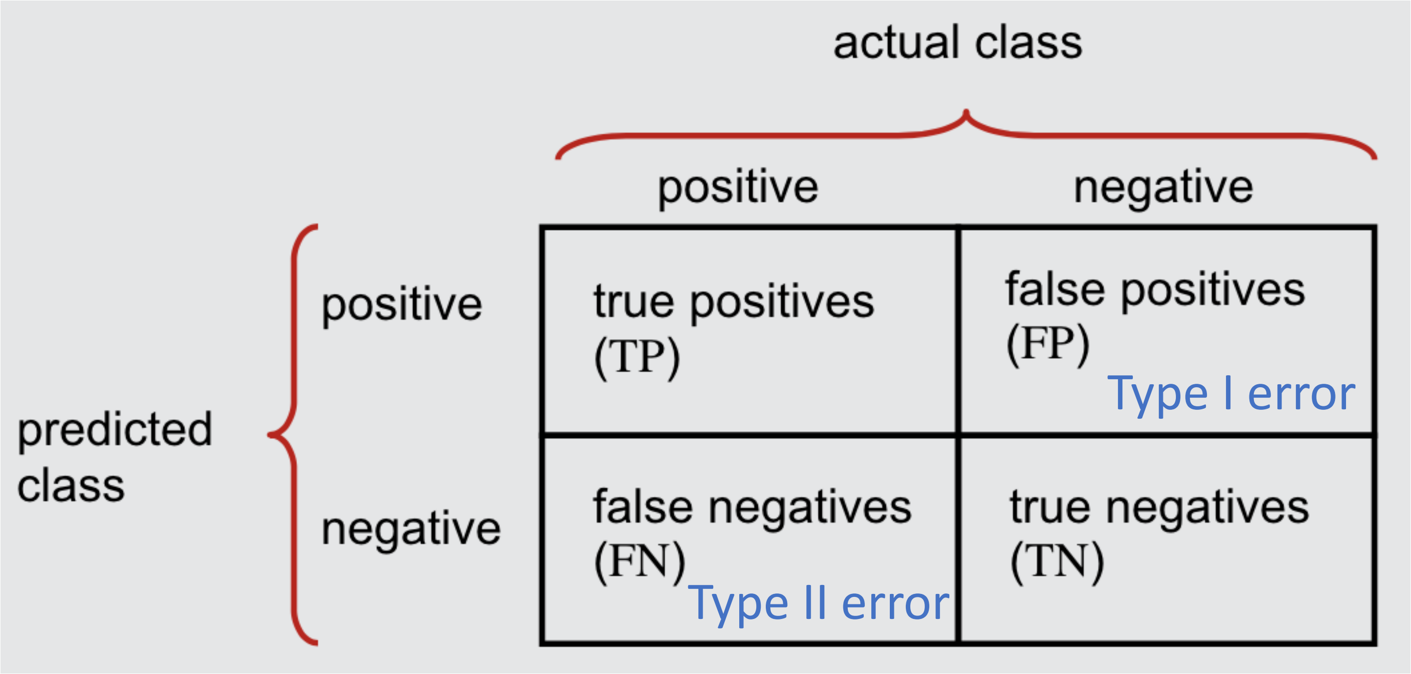
- True negative rate (TNR) = $\frac{\mathrm{TN}}{\mathrm{N}}$

  (specificity)

- False positive rate (FPR) = $\frac{\mathrm{FP}}{\mathrm{N}}$

- False negative rate (FNR) = $\frac{\mathrm{FN}}{\mathrm{P}}$

- Precision = $\frac{\mathrm{TP}}{\mathrm{P}-called} = \frac{P(\hat{y}=+1, y=+1)}{P(\hat{y}=+1)}$, $\mathrm{P}-called = \mathrm{TP}+\mathrm{FP}$



		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP) Type I error
	negative	false negatives (FN) Type II error	true negatives (TN)

P = TP + FN N = FP + TN

Applications:

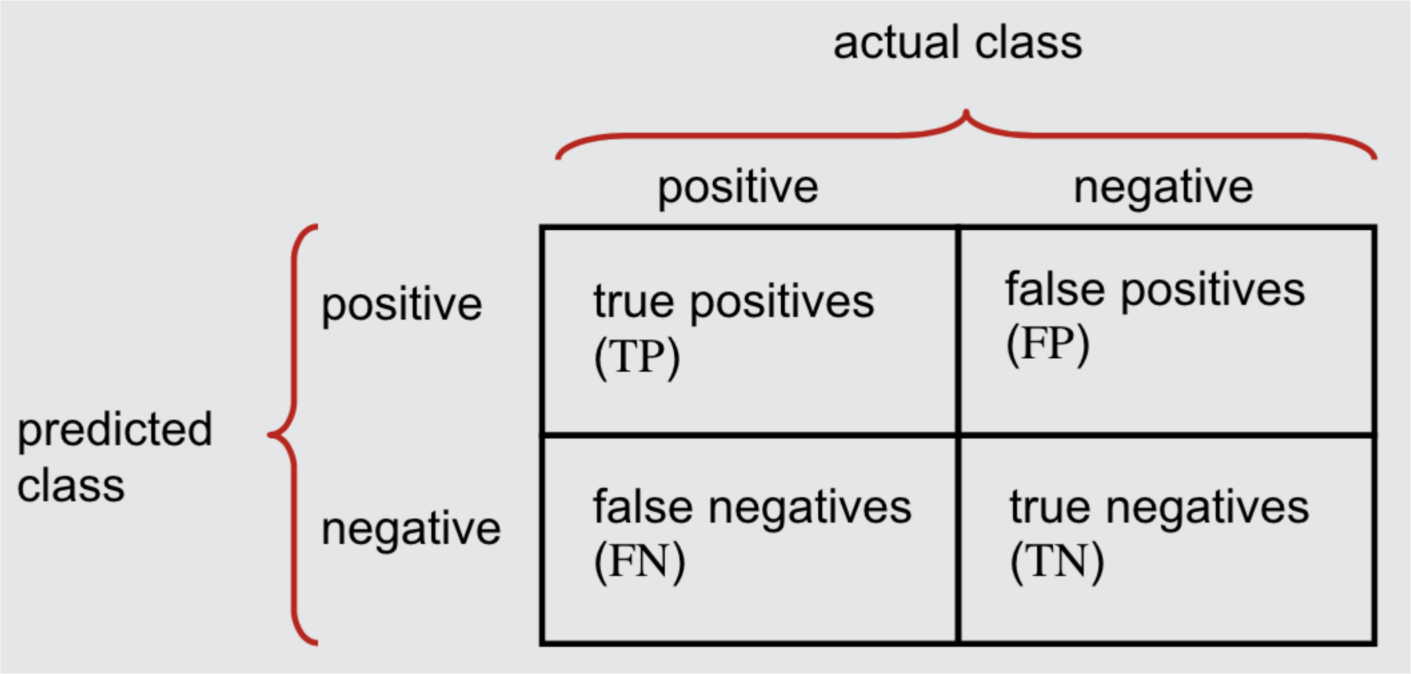
- Search engine: precision & recall
- Cancer classification: FNR vs. FPR

Adjusting TP, FP, TN, FN via thresholding

- Decision values (classification scores)

$c(x_i)$	y_i
.99	+
.98	+
.72	-
.51	-
.24	+

- $h_t(x) = \begin{cases} +1, & c(x) > t \\ -1, & c(x) \leq t \end{cases}$
- Choice of threshold t:
  - $t = +\infty$: $h_t \equiv -1 \Rightarrow \text{TPR} = 0,\ \text{FPR} = 0$
  - $t = 0$: $h_t \equiv +1 \Rightarrow \text{TPR} = 1,\ \text{FPR} = 1$

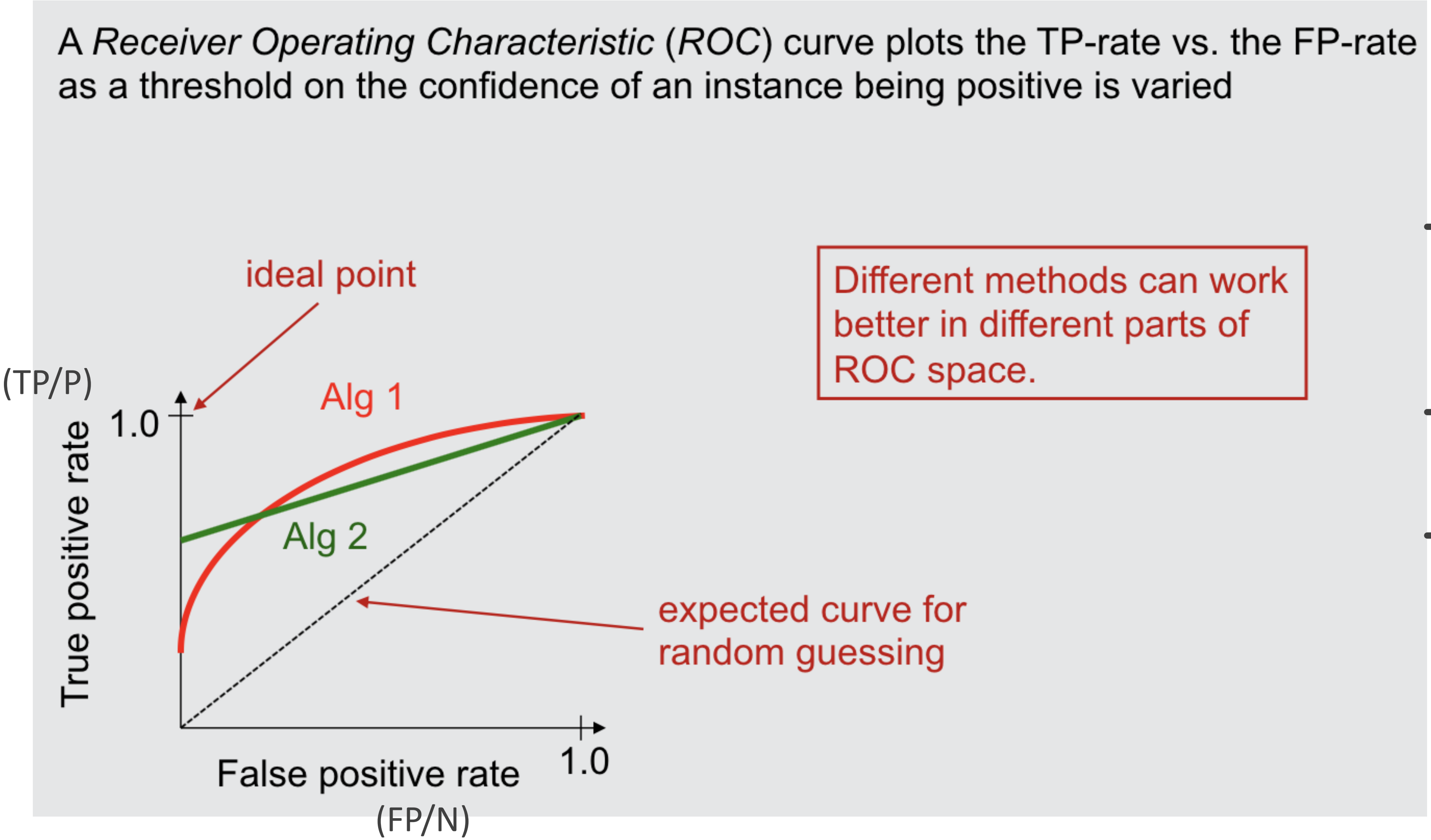


		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

P = TP + FN N = FP + FN

$$\boxed{\text{TPR} = \frac{\text{TP}}{\text{P}}, \quad \text{FPR} = \frac{\text{FP}}{\text{N}}}$$

ROC curve



A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied

Different methods can work better in different parts of ROC space.

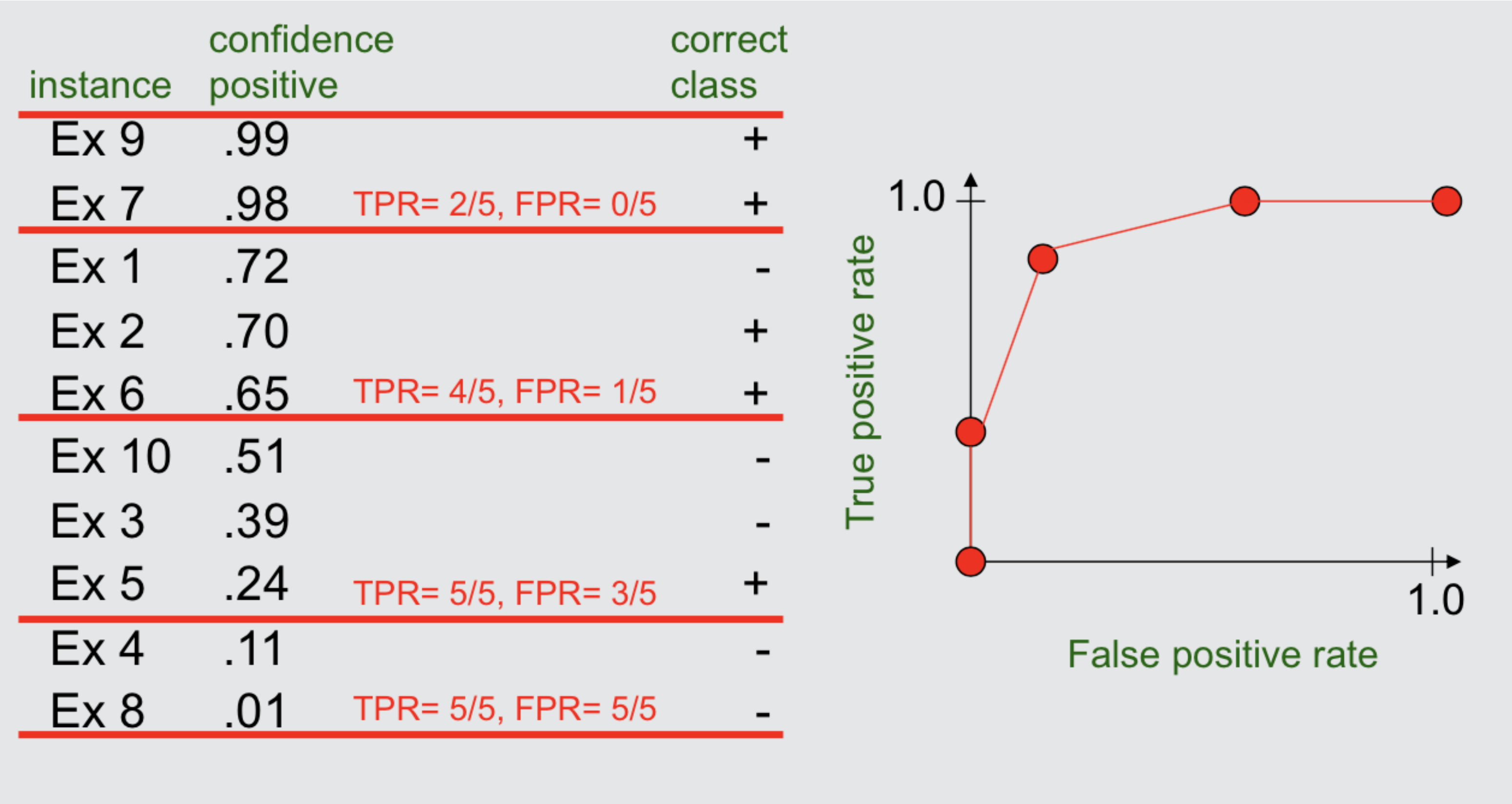
$c(x_i)$	y_i
.99	+
.98	+
.72	+
.51	-
.24	-

ROC curve

- Conceptually, consider every possible threshold, put a dot for each, and connect them.

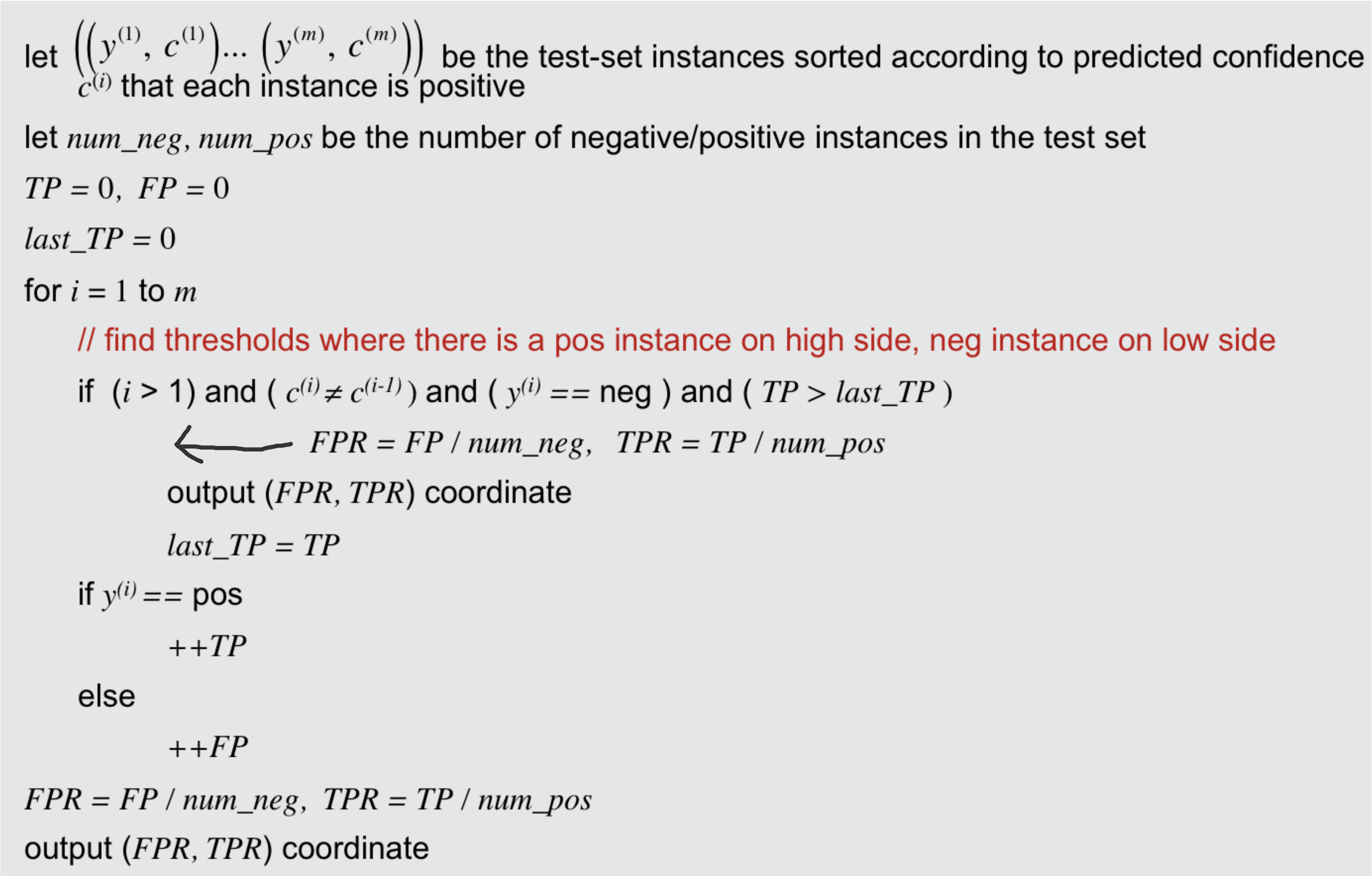
- Actually, just need to care about when the 'correct class' changes
  - results in staircase shape, but diagonal line can still happen.

- A popular alternative: just plot when going from + to -.
  (what's shown here)

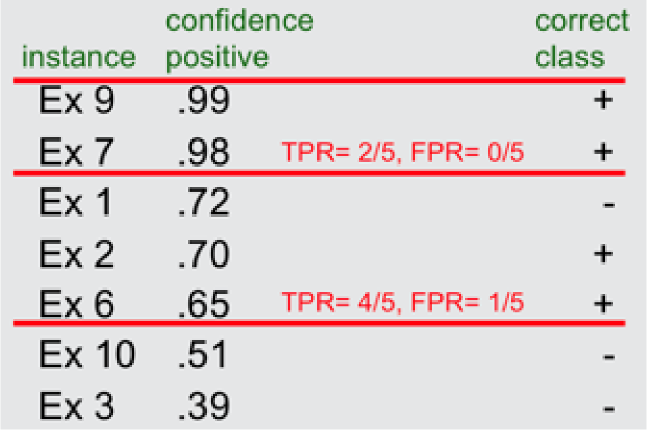


instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-

Calculating ROC curve

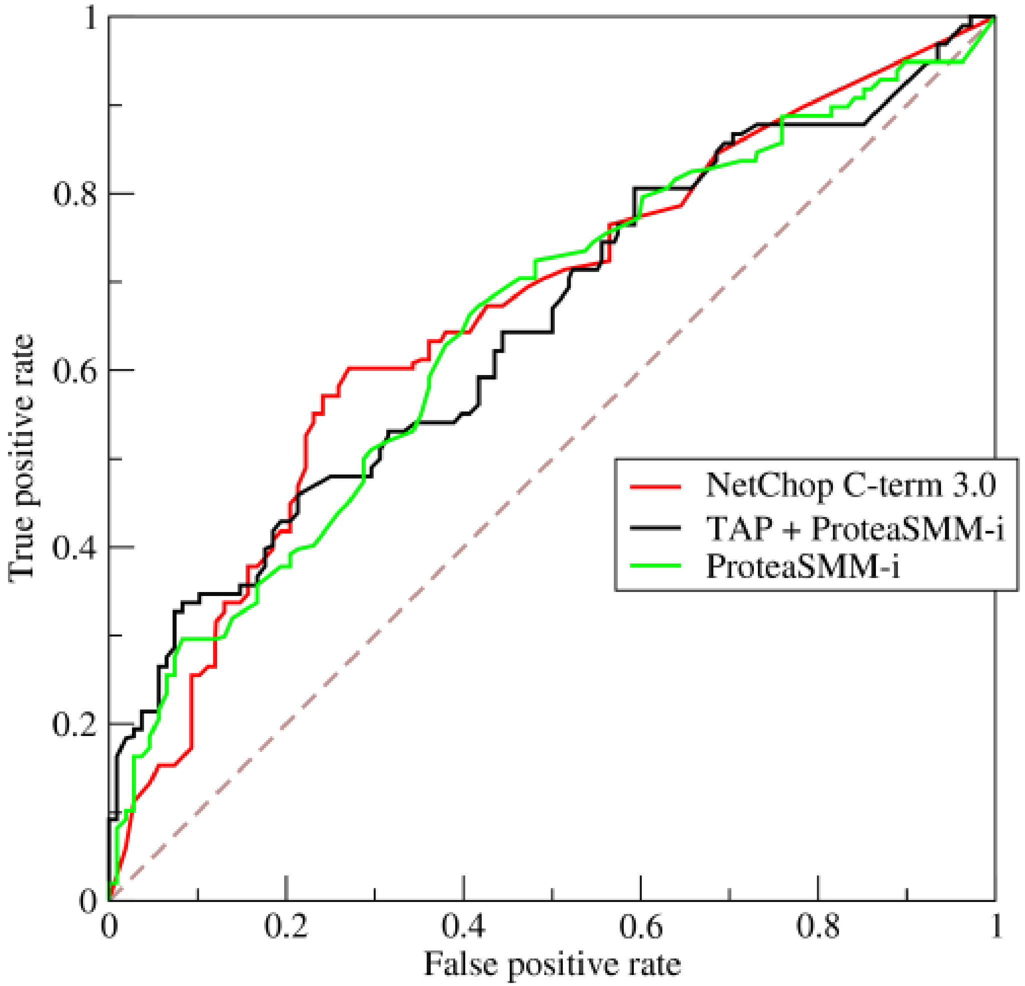


```
let $\left(\left(y^{(1)}, c^{(1)}\right) \ldots \left(y^{(m)}, c^{(m)}\right)\right)$ be the test-set instances sorted according to predicted confidence
    $c^{(i)}$ that each instance is positive
let $num\_neg$, $num\_pos$ be the number of negative/positive instances in the test set
$TP = 0,\ FP = 0$
$last\_TP = 0$
for $i = 1$ to $m$
    // find thresholds where there is a pos instance on high side, neg instance on low side
    if $(i > 1)$ and ( $c^{(i)} \neq c^{(i-1)}$ ) and ( $y^{(i)}$ == neg ) and ( $TP > last\_TP$ )
        ← $FPR = FP / num\_neg,\ \ TPR = TP / num\_pos$
        output $(FPR, TPR)$ coordinate
        $last\_TP = TP$
    if $y^{(i)}$ == pos
        $++TP$
    else
        $++FP$
$FPR = FP / num\_neg,\ \ TPR = TP / num\_pos$
output $(FPR, TPR)$ coordinate
```



instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-

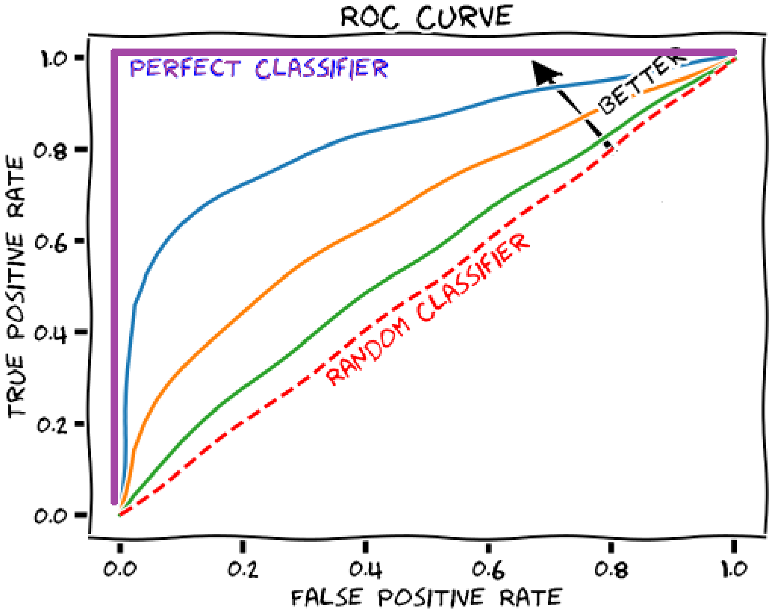
ROC curve examples



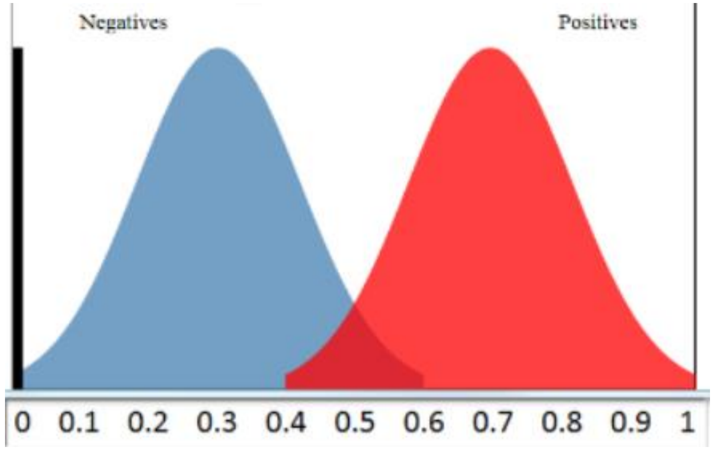
from Wikipedia

Area under ROC curve

- The boss says “could you just give me one number?”
- **AUC**: Area Under the ROC curve
  - Perfect scorer: AUC = 1
  - Random scorer: AUC = 0.5



- Quiz: Why does the perfect scorer c have a ROC that is ‘Γ’ shaped ?
  - What would a ``worst’’ ROC look like?



- Interpretation: “how well does c distinguish between + and -?”

https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

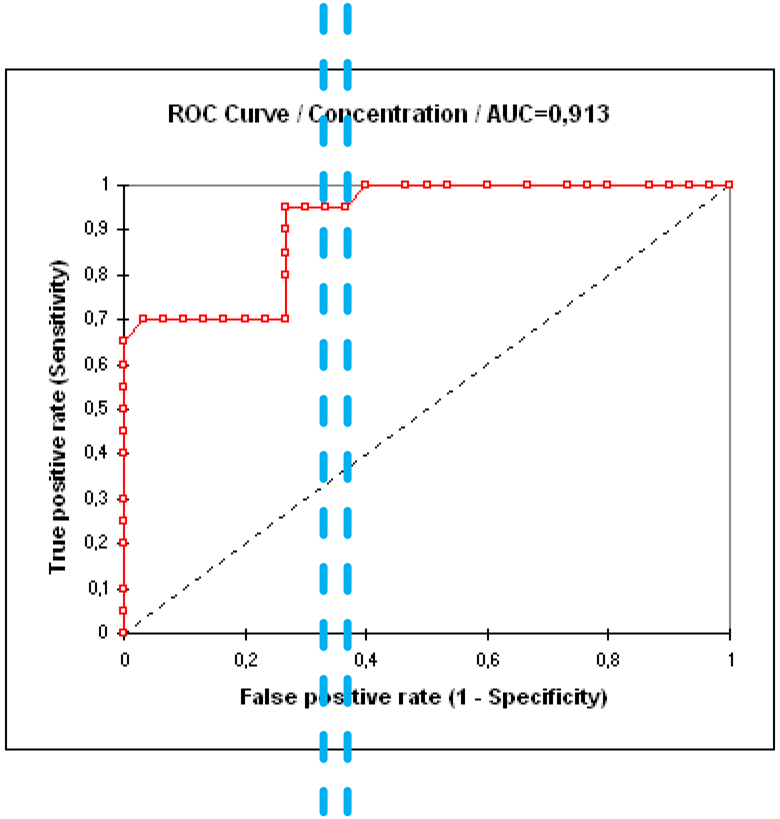
Area under ROC curve

- **Fact:** AUC of scorer c has the following formula:

$$\mathrm{AUC}(c) = \frac{\sum_{(x_-,-1)\in S_-} \sum_{(x_+,+1)\in S_+} I(c(x_+)>c(x_-))}{N_- \cdot N_+}$$

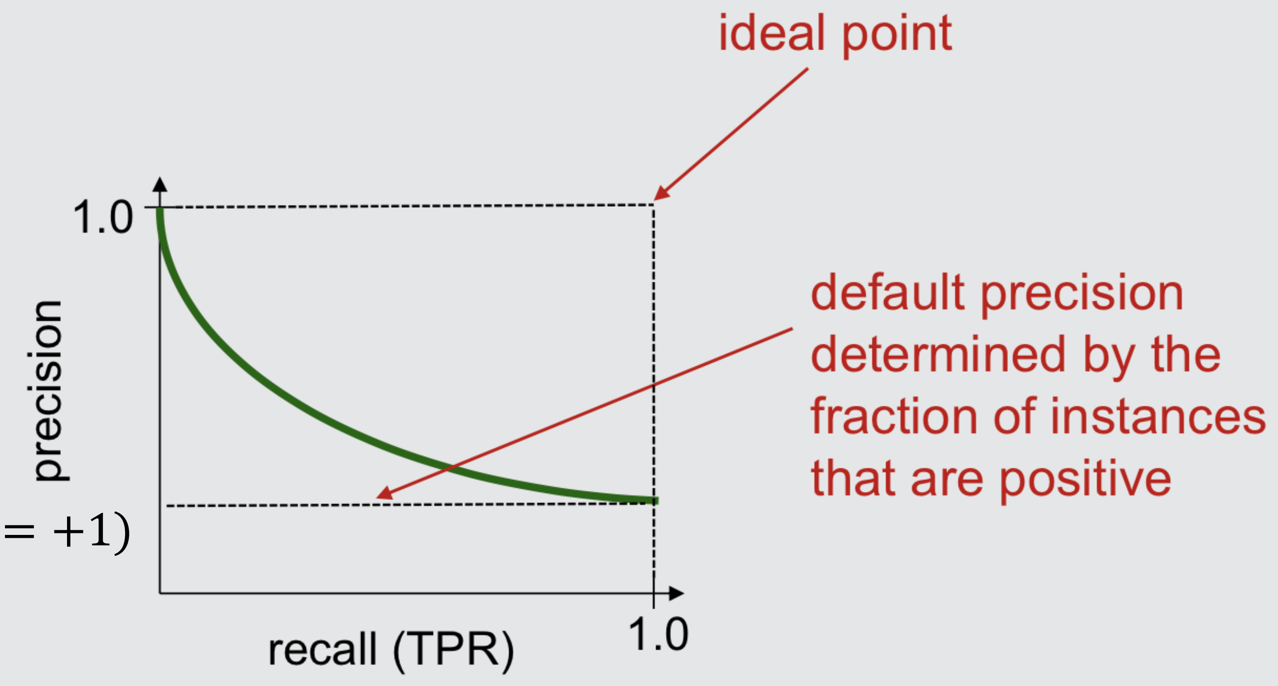
- Idea: the vertical slice corresponding to x_- has area

$$\frac{1}{N_-} \cdot \frac{\sum_{(x_+,+1)\in S_+} I(c(x_+)>c(x_-))}{N_+}$$



Precision-Recall (PR) curve

A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied



$P(y = +1 \mid \hat{y} = +1)$

$P(\hat{y} = +1 \mid y = +1)$

$c(x_i)$	y_i
.99	+
.98	+
.72	+
.51	-
.24	-

- This is *usually* a trade-off curve (not always): $t \downarrow \Rightarrow \text{recall} \uparrow,\ \text{precision usually} \downarrow$

PR-curve example

predicting patient risk for VTE

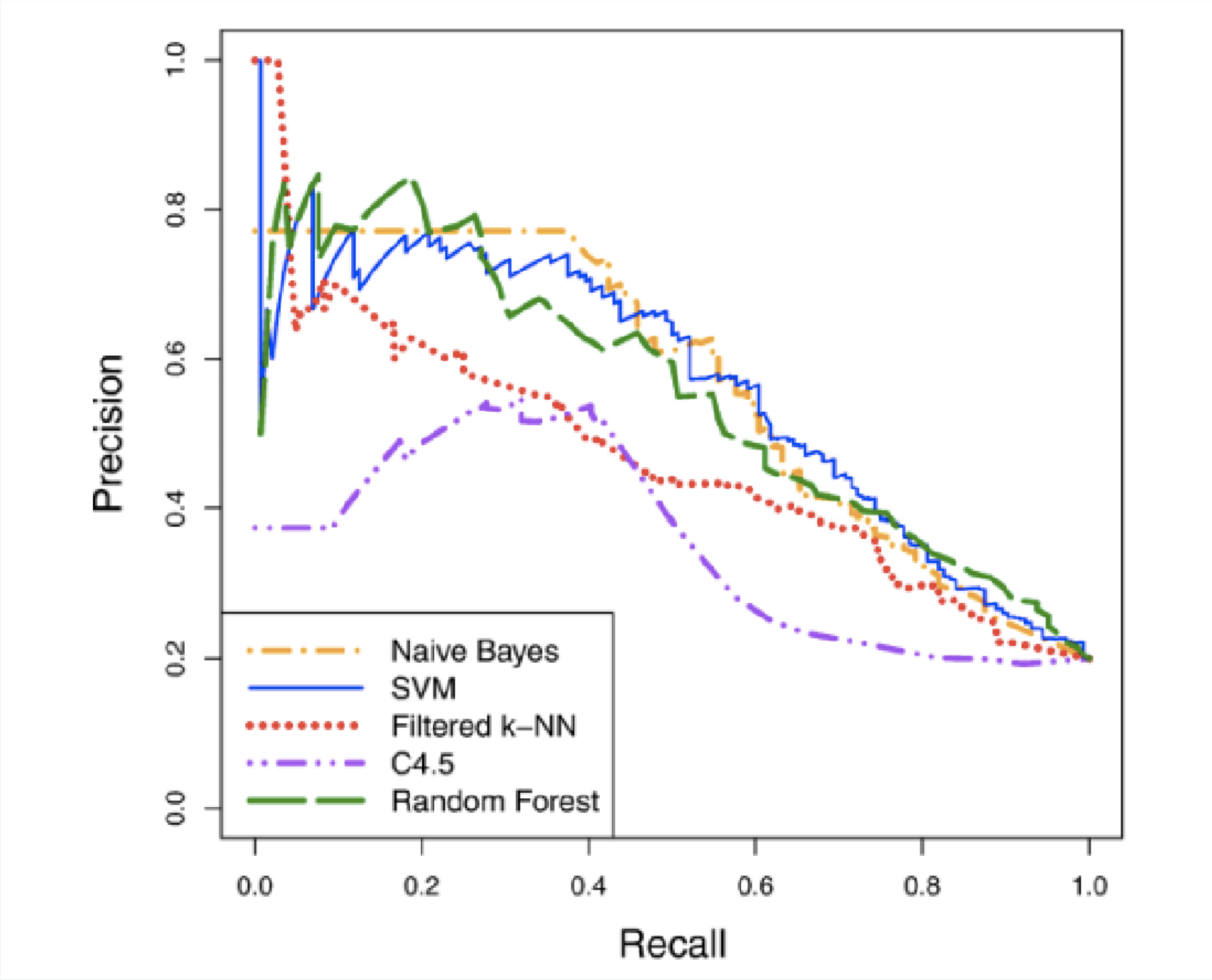


figure from Kawaler et al., *Proc. of AMIA Annual Symosium*, 2012

Summary of precision-recall

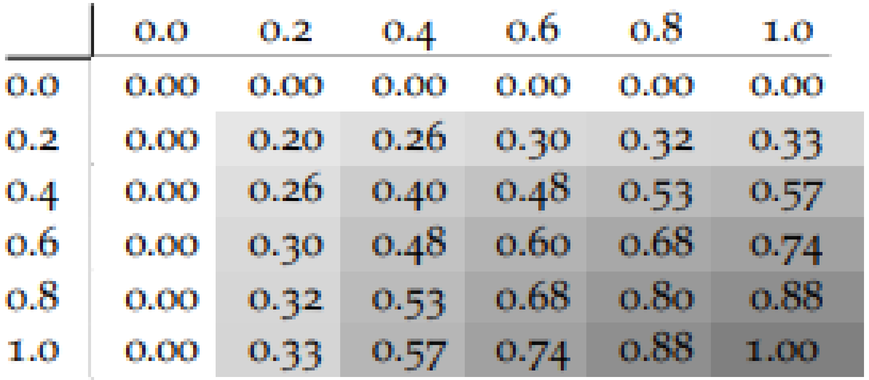
- Reporting one number
- Take the harmonic mean: **F1 score**

- Fact: minimum of two numbers <= harmonic mean <= geometric mean <= arithmetic mean

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}$$

- Emphasizes the smaller measure
  - E.g. $\mathrm{recall} = 0.1, \mathrm{precision} = 0.9 \Rightarrow \ F_1 = 0.18$

- Area under PR-curve is also a popular metric

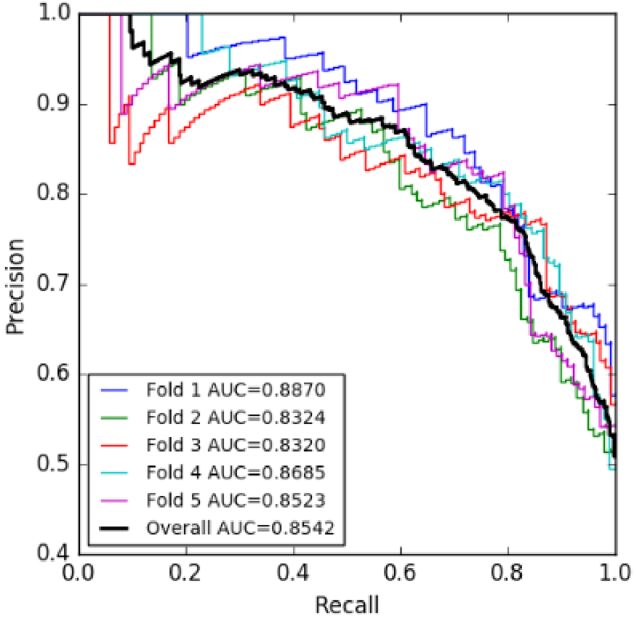


	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.00	0.20	0.26	0.30	0.32	0.33
0.4	0.00	0.26	0.40	0.48	0.53	0.57
0.6	0.00	0.30	0.48	0.60	0.68	0.74
0.8	0.00	0.32	0.53	0.68	0.80	0.88
1.0	0.00	0.33	0.57	0.74	0.88	1.00

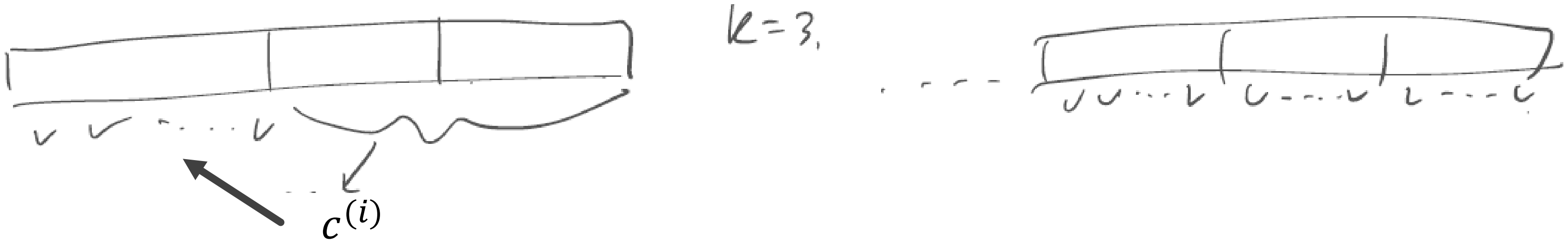
Table 5.2: Table of f-measures when varying precision and recall values.

How to plot ROC/PR curve when training set is small?

- k-fold cross-validation (CV):
  - Obtain k curves and plot them all



- **Pooled prediction:** compute c(x) *for all training examples* using k-fold CV.



Model Evaluation & Comparison

Motivation: evaluating & comparing ML models

Example

- Your ML model f has test set error = 6.9%
- Your nemesis, Gabe's, ML model g has test set error = 6.8%
- How confident are we to conclude that g has smaller population error than that of f?

- Intuition: We should be more (less) confident, if the test set is larger (smaller)
- Our uncertainty can be quantified with a *confidence interval*
- Determining the best model can be done rigorously with *hypothesis testing*

Disclaimer: we only focus on the key ideas (standard stats courses spend >= 5 lectures on this)

Confidence Intervals (CIs)

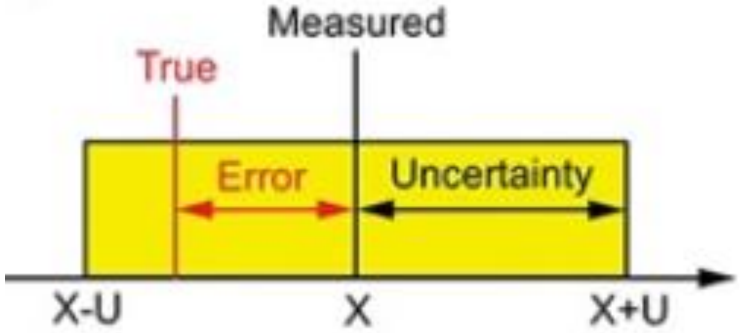
Intuition Find an interval such that we are *pretty sure* it encompasses the true parameter value (e.g. model accuracy).

Also commonly used when making measurements

Given data $X_1, \ldots, X_n$ and confidence $\alpha \in (0,1)$ find interval (a,b) such that,

$$P(\theta \in (a,b)) \geq 1 - \alpha$$

In English the interval (a,b) contains the true parameter value θ with probability **at least** $1 - \alpha$



- Allows us to quantify our *uncertainty* about θ (wider CI => higher uncertainty)
- Intervals must be computed from data $a(X_1, \ldots, X_n)$ and $b(X_1, \ldots, X_n)$
- Interval (a,b) is **random**, parameter θ is **not random** (it is fixed)
- Intuitively: α smaller => wider CI; n larger => narrower CI;

https://www.nelpretech.com/resources/measurement-uncertainty

Confidence interval: caveat in interpretation

- Suppose $\theta =$ true width of the window (in cm)

- By making 100 independent measurements, we have computed a 95% confidence interval: [247.5, 252.5]
- Does that mean:

$$\text{with probability } 0.95,\ \theta \in [247.5, 252.5] \qquad (*)$$

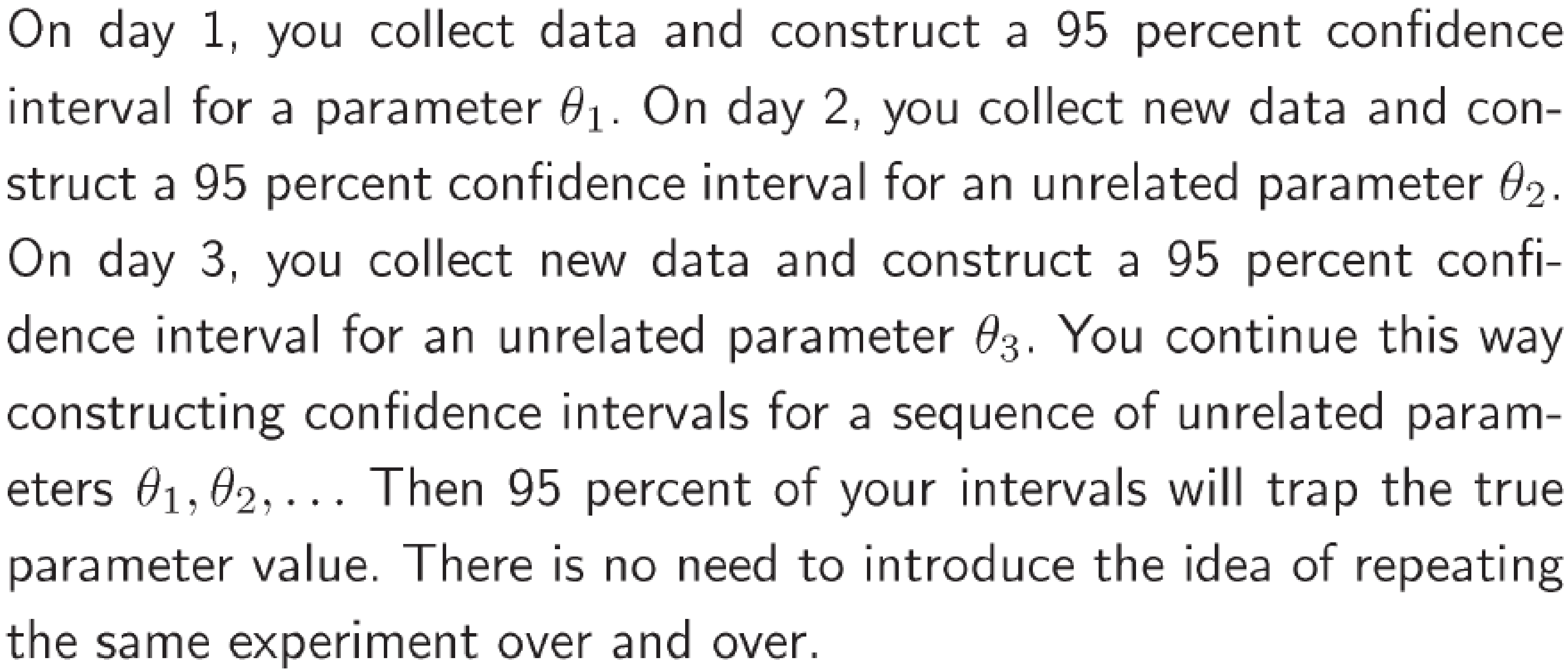
?

- **Hint**: What is the randomness in the above statement?

- **Answer:** The above interpretation is **incorrect** – “θ ∈[247.5, 252.5]” is either true or false – there is no randomness in this statement

- Then: what does $P(\theta \in (a(X), b(X))) \geq 0.95$ truly mean?

Interpretation



On day 1, you collect data and construct a 95 percent confidence interval for a parameter θ_1. On day 2, you collect new data and construct a 95 percent confidence interval for an unrelated parameter θ_2. On day 3, you collect new data and construct a 95 percent confidence interval for an unrelated parameter θ_3. You continue this way constructing confidence intervals for a sequence of unrelated parameters $\theta_1, \theta_2, \ldots$ Then 95 percent of your intervals will trap the true parameter value. There is no need to introduce the idea of repeating the same experiment over and over.

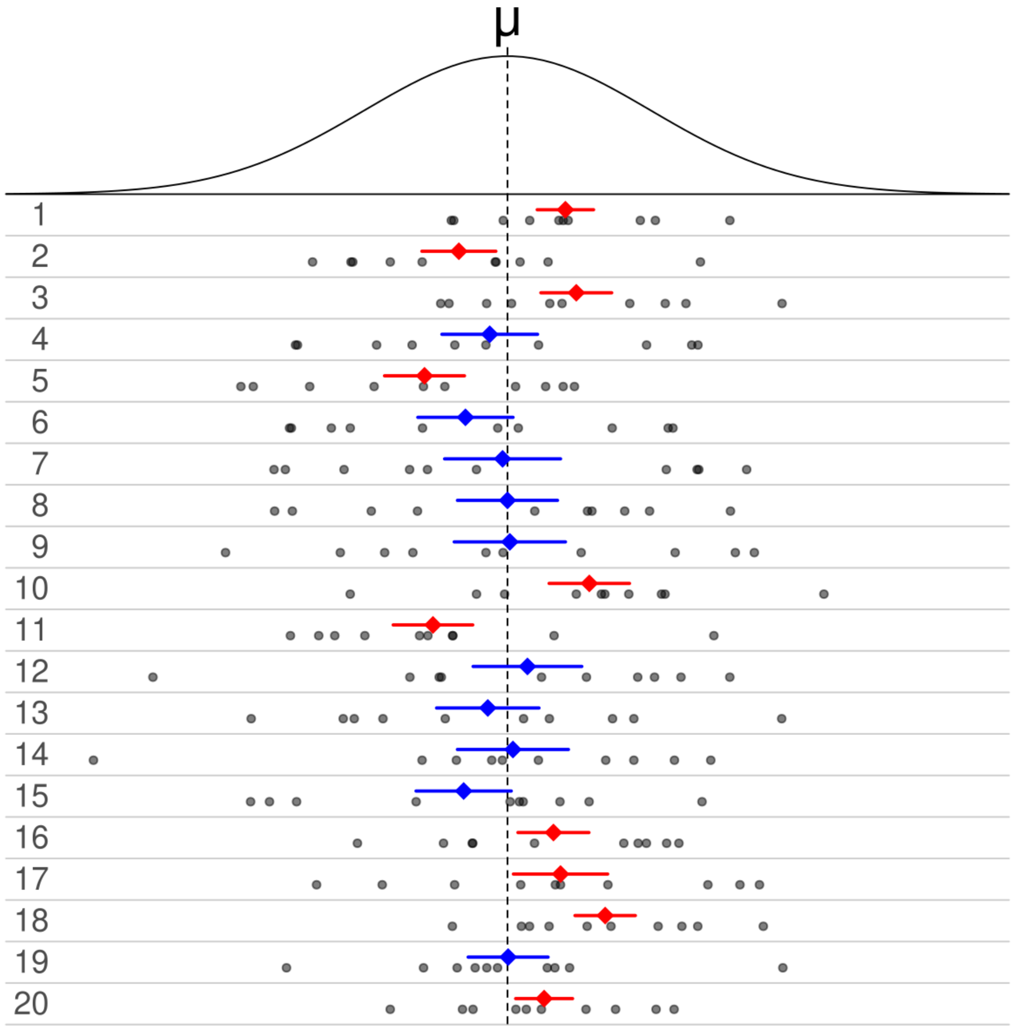
Another interpretation: as long as we collect a *representative* sample (which happens 95% of the time), its induced confidence interval contains θ

As long as we are not extremely unlucky, our induced confidence interval contains θ

[Source: Wasserman, L. 2004]

Knowledge Check

What is the confidence level of this confidence interval construction?



CSC 480/580 Principles of Machine Learning

05 Practical Considerations

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA®

Outline

- Classification metrics beyond error rate

- Model Evaluation & Comparison

- Debugging Learning Algorithms

- Bias / Variance Tradeoff

Confidence Intervals (CIs)

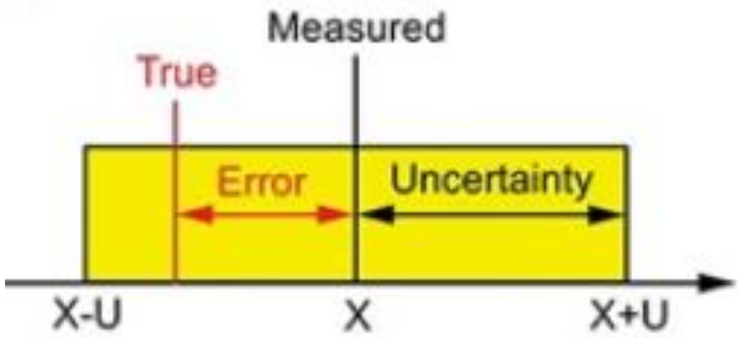
Intuition Find an interval such that we are *pretty sure* it encompasses the true parameter value (e.g. model accuracy).

Also commonly used when making measurements

Given data $X_1, \ldots, X_n$ and confidence $\alpha \in (0,1)$
find interval (a,b) such that,

$$P\underline{(\theta \in} (a,b)) \geq 1 - \alpha$$

In English the interval (a,b) contains the true parameter value θ with probability **at least** $1-\alpha$



- Allows us to quantify our *uncertainty* about θ (wider CI => higher uncertainty)
- Intervals must be computed from data $a(X_1, \ldots, X_n)$ and $b(X_1, \ldots, X_n)$
- Interval (a,b) is **random**, parameter θ is **not random** (it is fixed)
- Intuitively: α smaller => wider CI; n larger => narrower CI;

https://www.nelpretech.com/resources/measurement-uncertainty

CI construction: a standard recipe

- **Step 1:** Construct an estimator for θ based on S -- call it $\boxed{\hat{\theta}_S}$ (center of CI)
- **Step 2:** Let CI $I(S) \coloneqq [\hat{\theta}_S - w, \hat{\theta}_S + w]$, where w is chosen such that for all θ,

$$P\left(\theta \in [\hat{\theta}_S - w, \hat{\theta}_S + w]\right) \geq 1 - \alpha$$

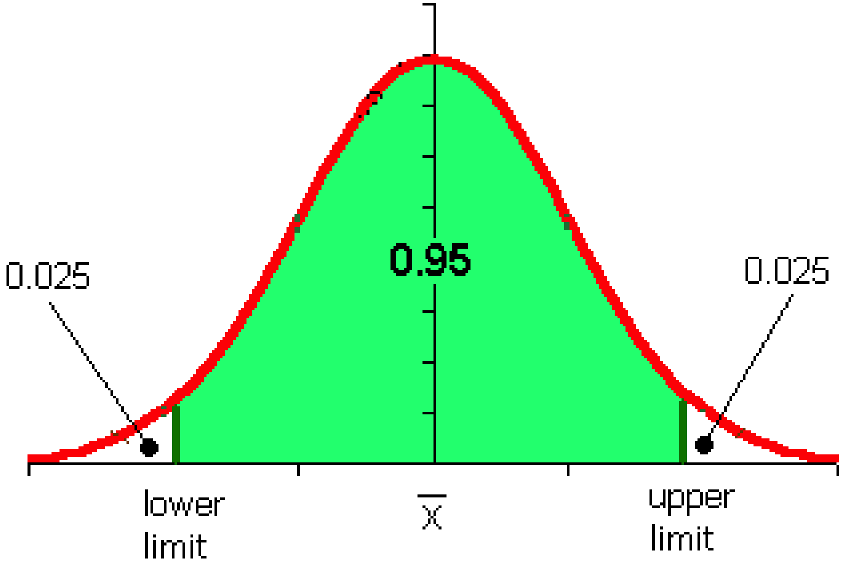
  - Equivalently,

$$P\left(\left|\theta - \hat{\theta}_S\right| \leq w\right) \geq 1 - \alpha$$

Important example: confidence interval for normal mean

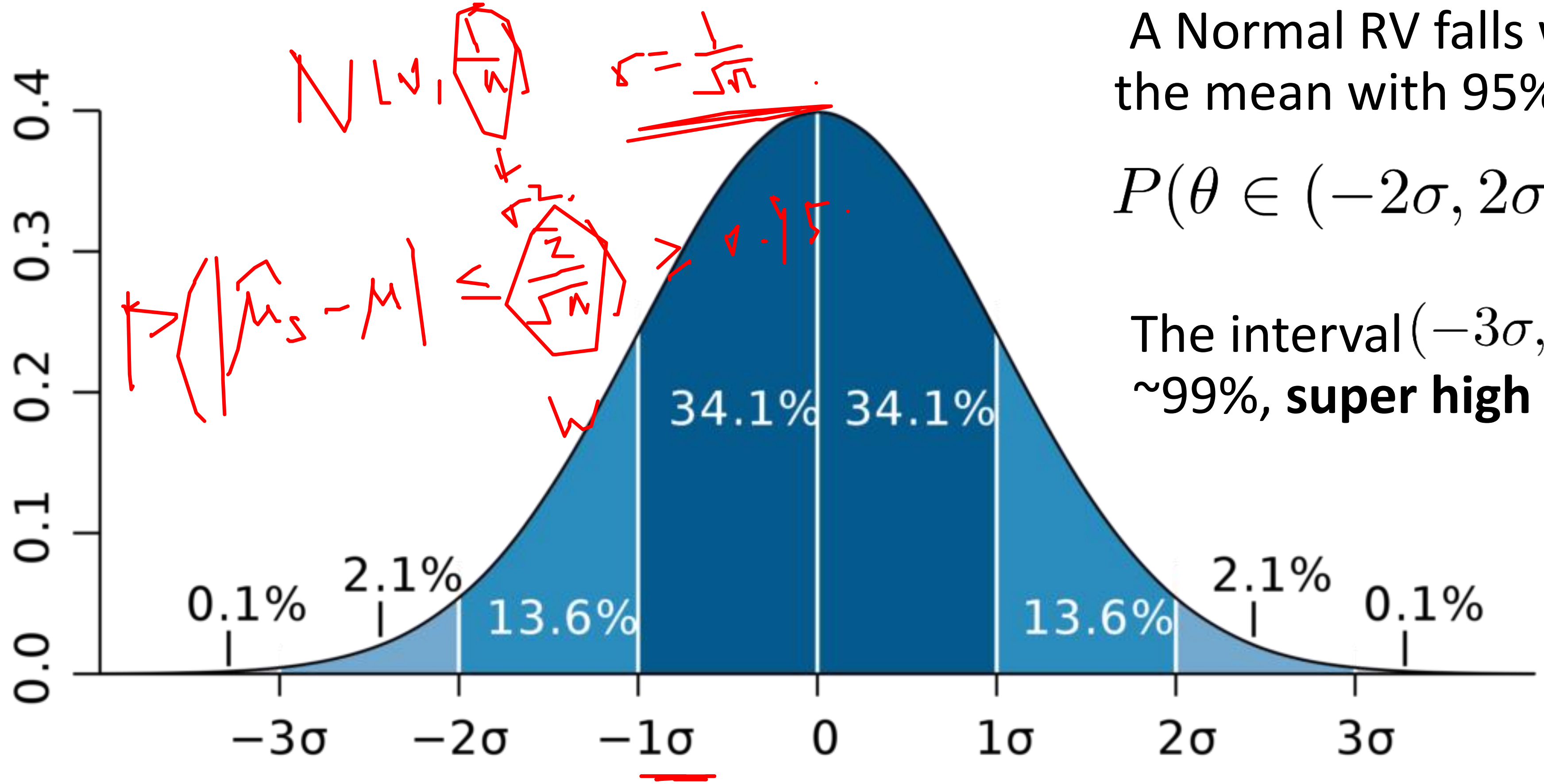
- $D_\mu = N(\mu, 1), S = (X_1, \dots, X_n) \sim D_\mu^n$
- Define $\hat{\mu}_S = \frac{1}{n}\sum_{i=1}^n X_i$
- $\boxed{\hat{\mu}_S - \mu} \sim N\left(0, \frac{1}{n}\right)$
- How to choose w such that $P(|\hat{\mu}_S - \mu| \leq \boxed{w}) \geq 1 - \alpha$?

Known variance



Confidence Intervals of the Normal Distribution

Many estimators follow a normal distribution with enough data (central limit theorem)



A Normal RV falls within $2\sigma$of the mean with 95% probability

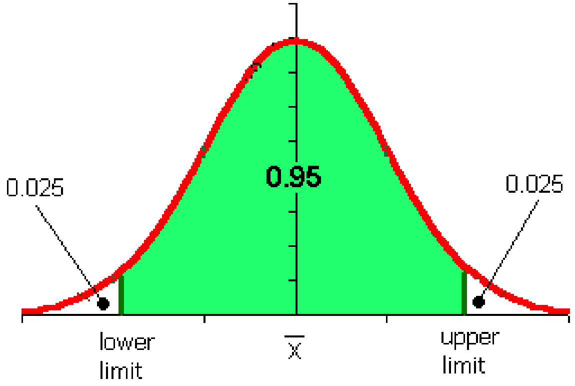
$$P(\theta \in (-2\sigma, 2\sigma)) \geq 0.95$$

The interval $(-3\sigma, 3\sigma)$ covers ~99%, **super high confidence**

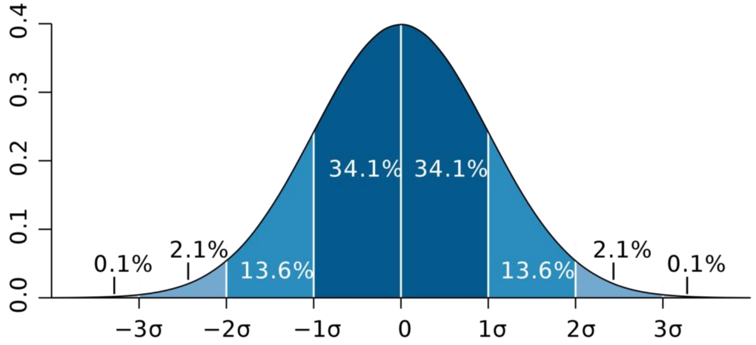
For various reasons, 95% has become standard confidence level

CI for normal mean (cont’d)

- $\hat{\mu}_S - \mu \sim N\left(0, \frac{1}{n}\right)$
- How to choose w such that $P(|\hat{\mu}_S - \mu| \leq w) \geq 1 - \alpha$?



- Note: $Z = \sqrt{n}\,(\hat{\mu}_S - \mu) \sim N(0,1)$ Central limit theorem
- Suffices to find z_α such that $P(|Z| \leq z_\alpha) \geq 1 - \alpha$, and let $w = \frac{z_\alpha}{\sqrt{n}}$



- Final $(1 - \alpha)$-confidence interval construction for μ: $I(S) = \left[\hat{\mu}_S - \frac{z_\alpha}{\sqrt{n}}, \hat{\mu}_S + \frac{z_\alpha}{\sqrt{n}}\right]$
- E.g. 95%-confidence interval for μ: $I(S) = \left[\hat{\mu}_S - \frac{1.96}{\sqrt{n}}, \hat{\mu}_S + \frac{1.96}{\sqrt{n}}\right]$

CI for means of general distributions, *unknown* variance

- Given D_θ with mean parameter θ with *unknown* variance
- $\hat{\sigma}_n^2 := \frac{\sum_{i=1}^n (X_i - \hat{\mu}_n)^2}{n-1} \Longrightarrow$ unbiased estimator of $\text{var}(D_\theta)$ σ^2

population

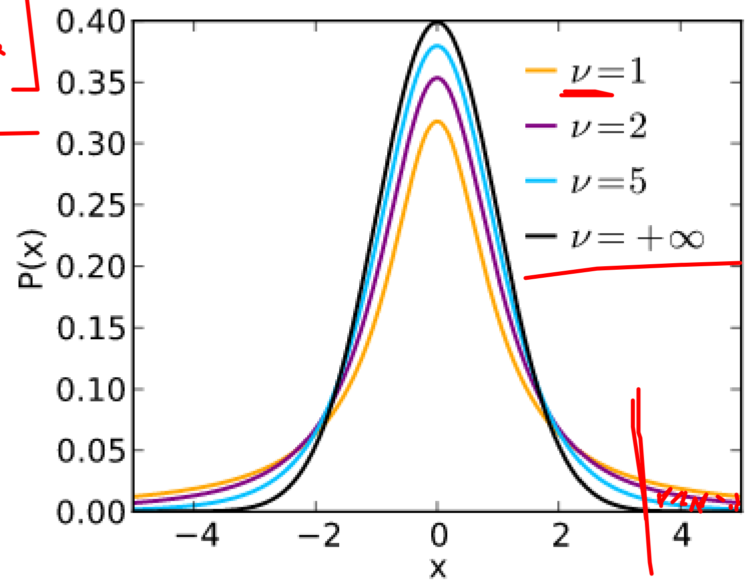
- **Fact**: Let $X_1, \dots, X_n \sim N(\mu, \sigma^2)$, and $\hat{\mu}_n := \frac{1}{n}\sum_{i=1}^n X_i$, then

$$\sqrt{n}\frac{\hat{\mu}_n - \mu}{\hat{\sigma}_n} \sim \text{student-t (mean 0, scale 1, degrees of freedom} = n-1)$$

- CI: $\left[\hat{\mu}_n \pm \frac{\hat{\sigma}_n t_\alpha}{\sqrt{n}}\right]$

$[-t_\alpha, t_\alpha]$

How do we estimate variance of algorithm performance?



```
import scipy.stats as st
alpha = 0.05
st.t.ppf(1-alpha/2,df=2)
=> 4.302652729911275

st.t.ppf(1-alpha/2,df=5)
=> 2.5705818366147395

st.t.ppf(1-alpha/2,df=10)
=> 2.2281388519649385

st.t.ppf(1-alpha/2,df=30)
=> 2.0422724563012373

st.t.ppf(1-alpha/2,df=100)
=> 1.9839715184496334
```

Bootstrapping CI

- General approach for constructing CI for *any statistic* (e.g. F1 score, AUROC, FPR, …)
- Goal: using sample S, estimate property h of D (:=$h(D)$) using confidence intervals

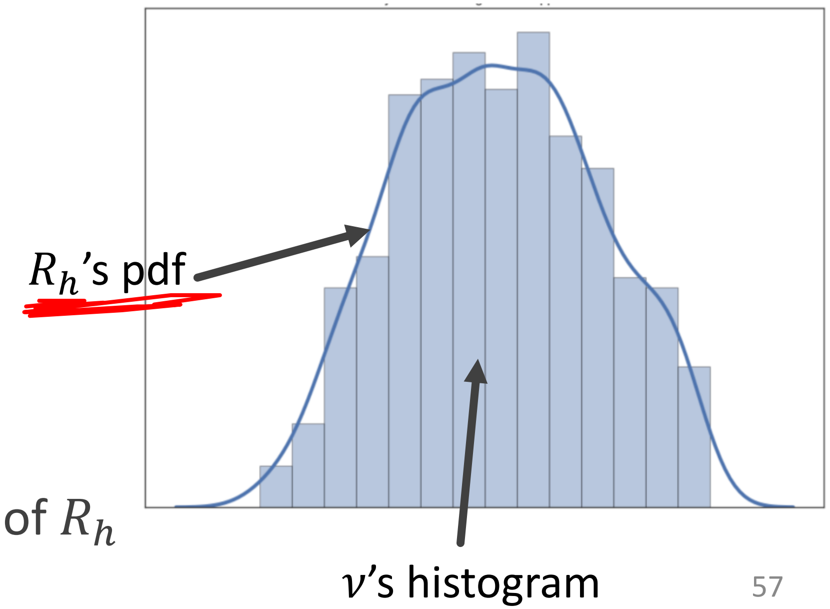
- **Step 1:** choose center of CI = $h(S)$
- **Step 2:** choose w_L, w_R such that $P(h(D) \in [h(S) - w_L, h(S) - w_R]) \geq 1 - \alpha$

Equivalently: $P(h(S) - h(D) \in [\underline{w_R}, \underline{w_L}]) \geq 1 - \alpha$

How?

- **Idea:** estimate the distribution of $h(S) - h(D)$, denoted by R_h

  by *bootstrapping* (resampling)
  - perform n times of "sampling with replacement" from S
  - repeat B times (e.g., B $= 100$) to obtain $S_1, \dots, S_B$
  - take $\underline{\nu} :=$ **empirical distribution** of $\{h(S_b) - h(S)\}_{b=1}^{B}$, as the **'shape'** of R_h



Bootstrapping CI: Implementation

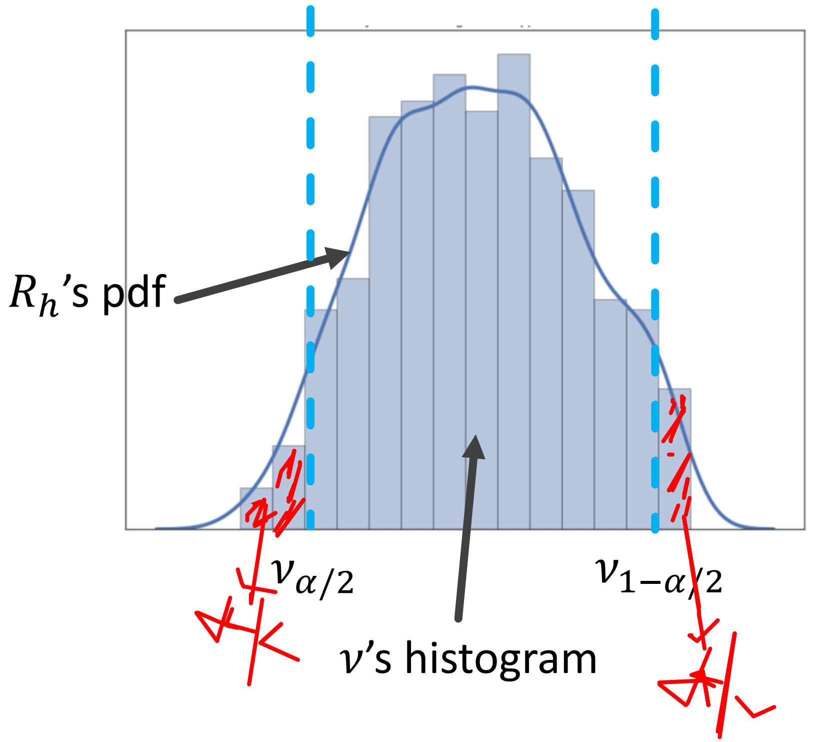
- Goal: choose w_R, w_L, s.t. $P(h(S) - h(D) \in [w_R, w_L]) \geq 1 - \alpha$

- ν (histogram of $\{h(S_b) - h(S)\}_{b=1}^{B}$) approximates R_h, the probability distribution of $h(S) - h(D)$

- Choose:

$$w_R = \nu_{\alpha/2}, w_L = \nu_{1-\alpha/2} \text{ -- quantiles of } \nu$$

Intuition: this well-approximates quantiles of R_h



- How to calculate ν's quantiles?
  - Sort $\{h(S_b) - h(S)\}_{b=1}^{B}$ in increasing order; say v[0..(B-1)]
  - $\nu_{1-\alpha/2}$:= the top 0.025 (i.e., v[int(0.975*B)])
  - $\nu_{\alpha/2}$:= the bottom 0.025 (i.e., v[int(0.025*B)])

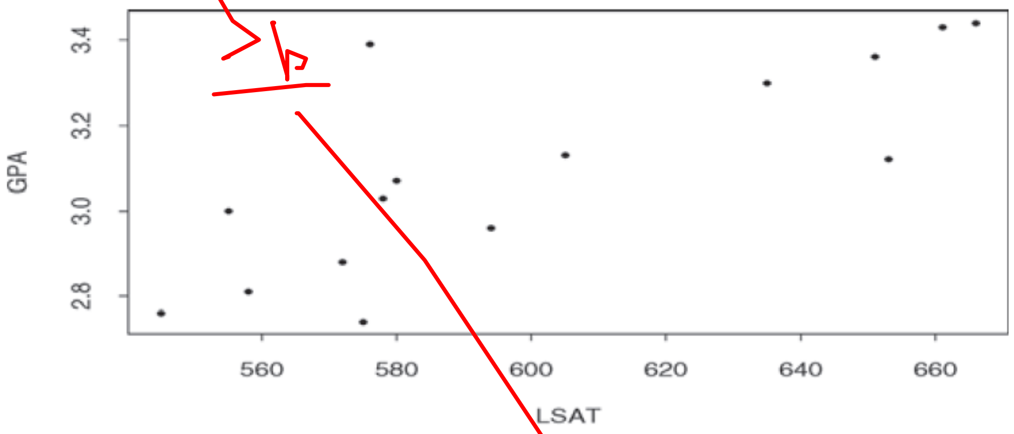
- **Final bootstrap CI:** $[h(S) - w_L, h(S) - w_R] = [h(S) - \nu_{1-\alpha/2}, h(S) - \nu_{\alpha/2}]$

Bootstrap Example

Example Suppose we have LSAT scores and GPA for 15 law students and wish to estimate the correlation between LSAT and GPA:

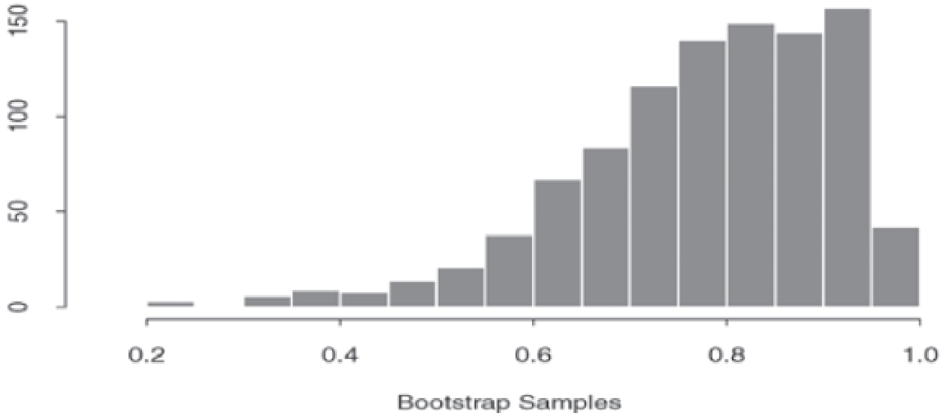
LSAT	576	635	558	578	666	580	555	661
	651	605	653	575	545	572	594	
GPA	3.39	3.30	2.81	3.03	3.44	3.07	3.00	3.43
	3.36	3.13	3.12	2.74	2.76	2.88	3.96	

$\Sigma \hat{\rho}$



95% Bootstrap confidence interval from B=1000 estimates of the **correlation**,

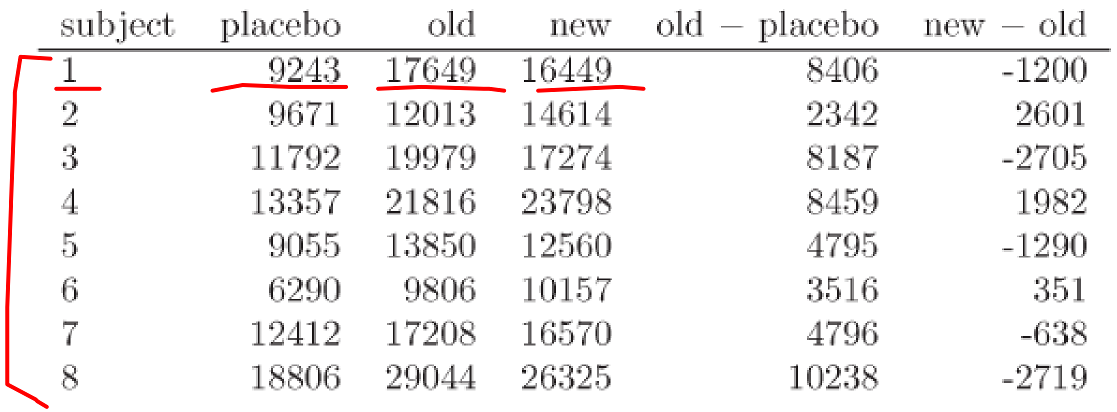
$$\underline{.78 \pm .274} \Rightarrow \boxed{(.51, 1.00)}$$



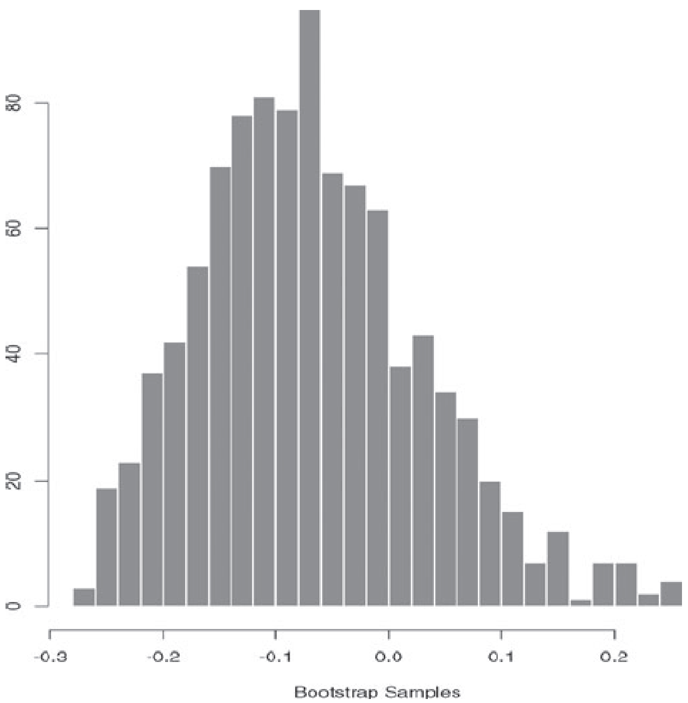
[Source: Wasserman, L. 2004]

Bootstrap Example

Eight subjects who used medical patches to infuse a hormone into the blood using three treatments: placebo, old-patch, new-patch



subject	placebo	old	new	old − placebo	new − old
1	9243	17649	16449	8406	-1200
2	9671	12013	14614	2342	2601
3	11792	19979	17274	8187	-2705
4	13357	21816	23798	8459	1982
5	9055	13850	12560	4795	-1290
6	6290	9806	10157	3516	351
7	12412	17208	16570	4796	-638
8	18806	29044	26325	10238	-2719



Estimate whether relative efficacy is the same under new drug,

$$\theta = \frac{\mathbf{E}[\text{new} - \text{old}]}{\mathbf{E}[\text{old} - \text{placebo}]}$$

Bootstrap B=1,000 samples yields 95% confidence interval,

$$\theta \in (-0.24, 0.15)$$

Hypothesis testing: motivation

- How to prove that your new system A is better than existing one B

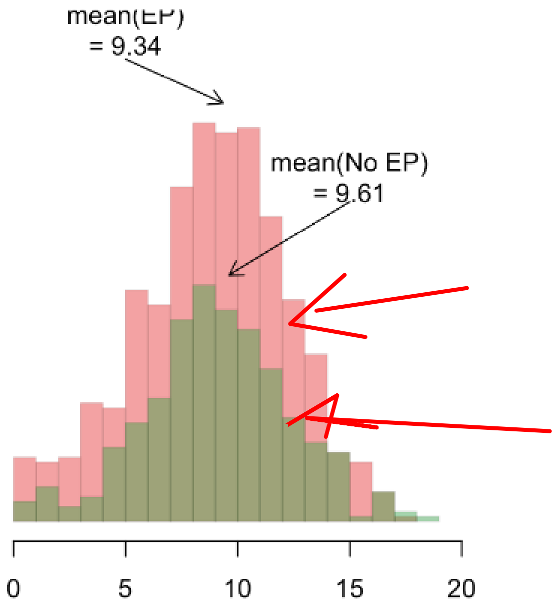
- **Ex (document retrieval):** A and B generate 10 lists of documents given 10 different keywords => the evaluator provides rating (1-5) for both lists

Evaluator	1	2	3	4	5	6	…
A	5	2	2	5	4	2	…
B	4	1	1	4	3	1	…

- **Ex (classification):** 100 test data points => Record A and B’s classification correctness in each

Two-sample hypothesis testing: definition

- Distribution family $\{D_\theta\}$ with parameter θ
- Samples $S_X = (X_1, \dots, X_n)$ and $S_Y = (Y_1, \dots, Y_n)$ drawn iid from distribution D_{θ_X} and D_{θ_Y}, respectively



- Equality test version:
  - Null hypothesis $H_0: \theta_X = \theta_Y$
  - Alternative hypothesis $H_1: \theta_X \neq \theta_Y$
- E.g. $D_\mu = \mathrm{N}(\mu, 1)$, $H_0: \mu_X = \mu_Y$
- Design hypothesis tester T such that:
  - T maps $S = (S_X, S_Y)$ to hypothesis index 0 or 1
  - Type-I error rate $P_{H_0}(T(S) = 1) \leq \alpha$ (α: significance level; say, 0.05)
  - Intuition: do not overturn the null hypothesis unless we have strong evidence – conservatism

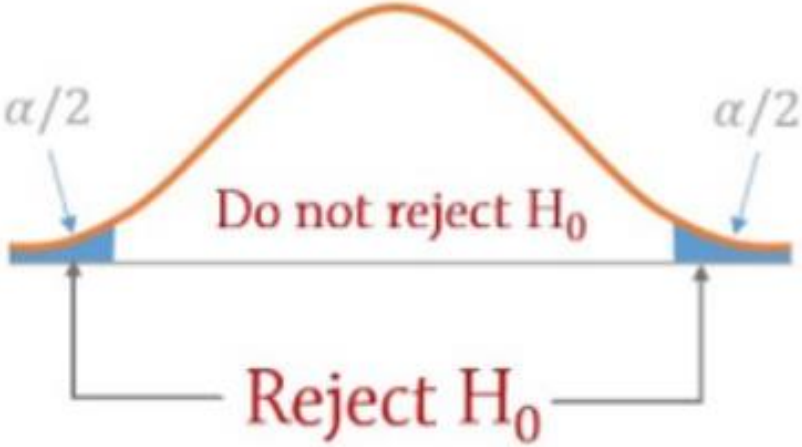
Paired t-test

- $S_X = (X_1, \dots, X_n)$ and $S_Y = (Y_1, \dots, Y_n)$ drawn iid from distribution $D_{\theta_X} = N(\mu_X, \sigma_X^2)$ and $D_{\theta_Y} = N(\mu_Y, \sigma_Y^2)$, respectively
    - $H_0: \mu_X = \mu_Y$
    - $H_1: \mu_X \neq \mu_Y$
- Here, $\mu_X, \mu_Y, \sigma_X^2, \sigma_Y^2$ are all unknown

Evaluator	1	2	..
A	5	2	..
B	4	1	..

An intuitive approach:

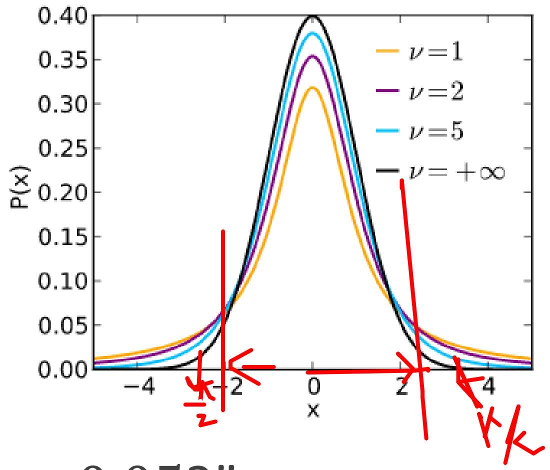
- Let $\delta_i := X_i - Y_i$, for all $i = 1, \dots, n$
- Let $\bar{\delta}_n := \frac{1}{n}\sum_{i=1}^{n} \delta_i$



- Intuition: reasonable to accept H_0 if $|\bar{\delta}_n| \leq c$, for some $c > 0$
- Design hypothesis test T so that $P_{H_0}(T(S) = 1) \leq \alpha$ => c needs to be set large enough
- Challenge: $\bar{\delta}_n$'s distribution not known, since σ_X^2, σ_Y^2 are unknown

Paired t-test

- **Observe:** Under H_0, $\delta_i \sim N(0, \sigma^2)$, $i = 1, \dots, n$, where $\sigma^2 = \sigma_X^2 + \sigma_Y^2$

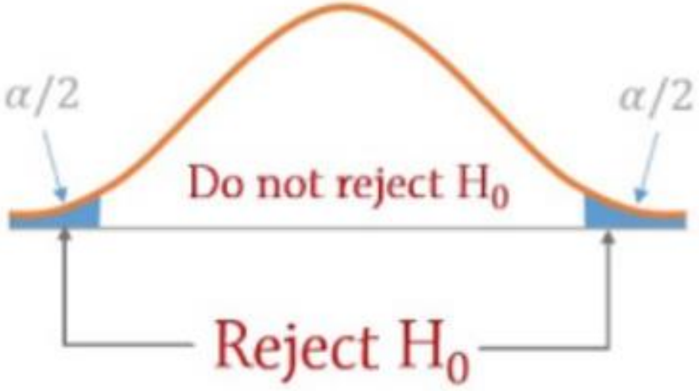


- Recall **Fact**: Let $\delta_1, \dots, \delta_n \sim N(0, \sigma^2)$, and $\bar{\delta}_n := \frac{1}{n}\sum_{i=1}^{n} \delta_i$, $\hat{\sigma}_n^2 := \frac{\sum_{i=1}^{n}(\delta_i - \bar{\delta}_n)^2}{n-1}$

$$Z = \boxed{\sqrt{n}\frac{\bar{\delta}_n}{\hat{\sigma}_n}} \sim \text{student-t (mean 0, scale 1, degrees of freedom} = n-1)$$

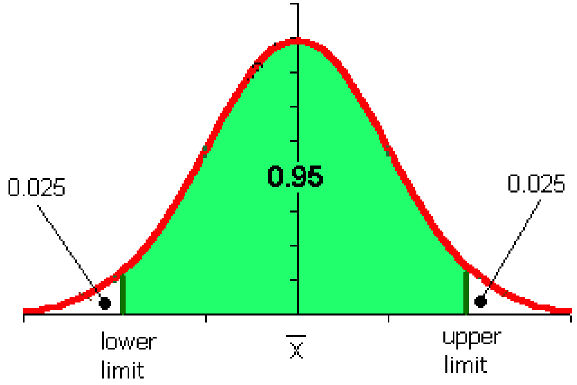
- Let's ask "under H_0, what is a plausible range of values of Z with failure rate $\alpha = 0.05$?"
  - Find the 0.025, 0.975-quantiles of Z => $t_{0.025}, t_{0.975}$
  - Hypothesis tester

$$T(S) = I(Z \notin [t_{0.025}, t_{0.975}]) = I\left(\sqrt{n}\frac{\bar{\delta}_n}{\hat{\sigma}_n} \notin [t_{0.025}, t_{0.975}]\right)$$



Hypothesis testing: additional remarks

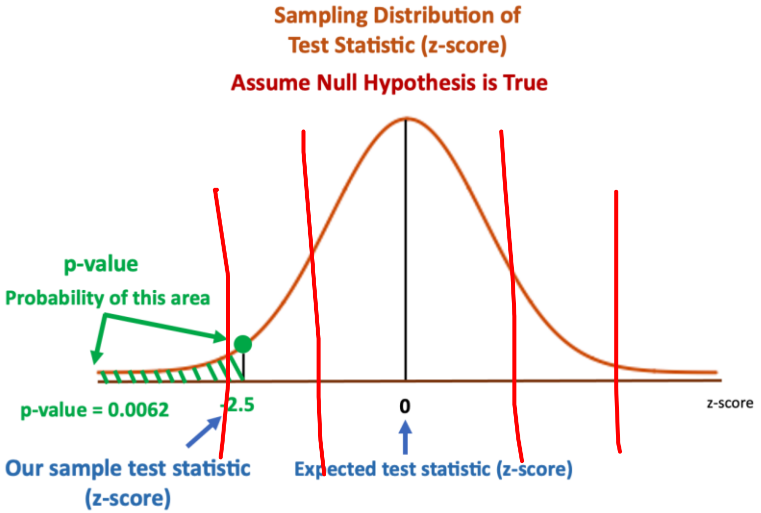
- Confidence intervals can be used for hypothesis testing
  - $S = (X_1, \dots, X_n)$ drawn iid from distribution D_μ
  - $H_0 : \mu = 0$
  - $H_1 : \mu \neq 0$
  - I is a $(1-\alpha)$-CI construction for μ => hypothesis test $T(S) = I(0 \notin I(S))$ has significance α



- p-value: given dataset S, and a family of hypothesis tests T_α's with different significance level α's

  $p =$ the smallest α with which you can still reject H_0

  Smaller p-value => result more significant



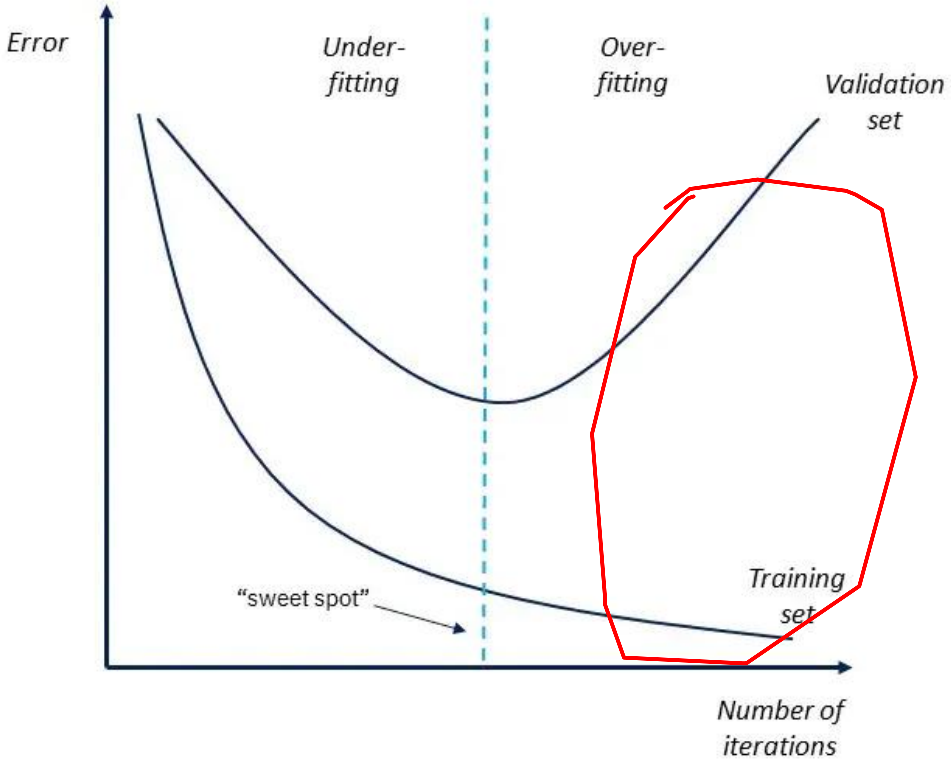
- Hypotheses should be decided *before* observing data
  - Otherwise results may be misleading (p-hacking)

Debugging Learning Algorithms

Debugging Learning Algorithms

Is the problem with generalization to test data?

- Is it doing well on training?
  - Unrealistic to do *better* on test than on training
  - If not, problem may be *representation*: need better features or better data
- If it does well on training then problem is *generalization*
  - Model may be too complicated (overfitting)
  - Too many features, not enough training data

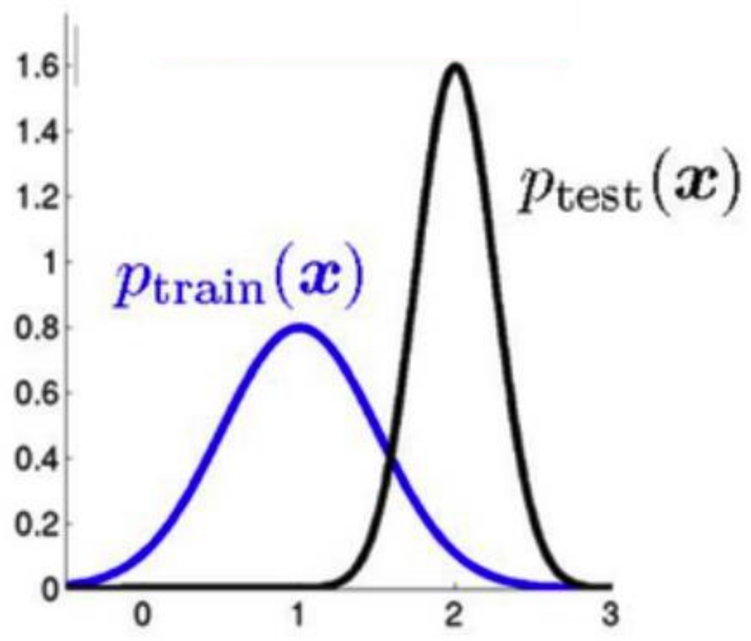


Debugging Learning Algorithms

Is there a mismatch between training and test?

- Training data may be inadequate

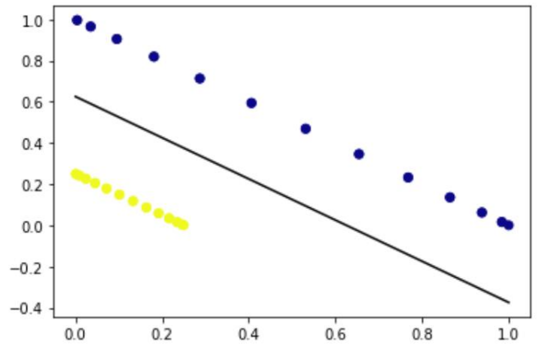
- If enough training & test data:
    - Do results change with different train / test split?
    - If so then test distribution is probably strange



Debugging Learning Algorithms

Is the learning algorithm implemented correctly?

- Is it optimizing the loss function that you intended?
- Try measuring / visualizing your loss function during training-is it going down?
- Do the data meet your algorithmic assumptions?
- Hand-craft datasets where you know the desired behavior
  - KNN on XOR function
  - Perceptron on data that is trivially linearly-separable
  - Decision tree on axis-aligned data
  - Generally, create dataset that meets assumptions of your algorithm



Debugging Learning Algorithms

Do you have adequate representation?

- Your feature set could be inadequate
- For binary classification try this…
  - Add a feature (maybe call it `CheatingIsFun`)
  - Set value to +1 for positive instances and -1 for negative instances
  - This feature is a *perfect indicator*-problem is now trivially solvable
  - Does your algorithm solve it?
- If your algorithm doesn’t get near 0% error then you *may* have a bug! (or more data / less features)
- If it does then you need better features or a different model (e.g. decision tree vs. linear model)

Debugging Learning Algorithms

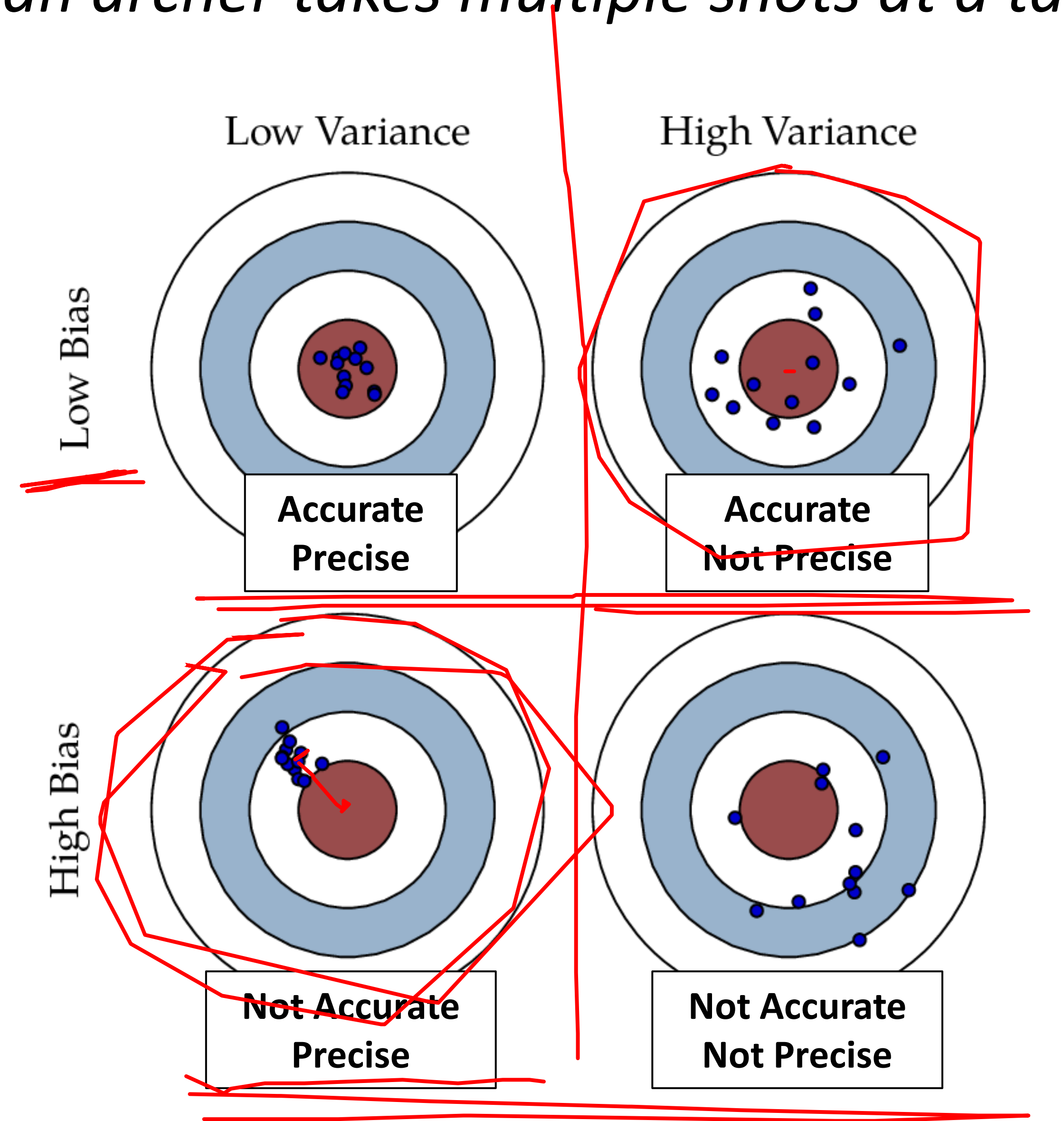
Do you have enough data?

- Conventional wisdom: Always have *at least* as many training data as you have learnable model parameters
  - Caveat: not required in modern overparameterized setting
- Try training on 80% of your training data
  - Does performance suffer?
  - How much? A lot?
  - If so then getting more data is likely helpful
  - If not then you may be *data saturated*-look elsewhere
- More training data should never lead to *worse* performance (just slower training)

Bias / Variance Tradeoff

Bias-Variance Tradeoff

Suppose an archer takes multiple shots at a target…



Bias-Variance Tradeoff

Is an unbiased estimator "better" than a biased one? It depends…

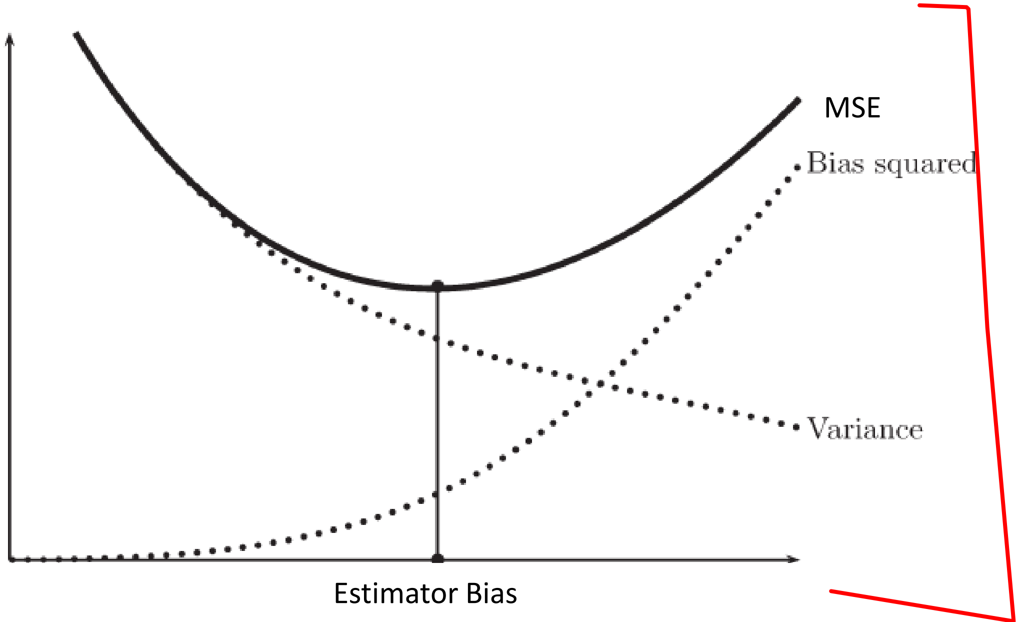
Evaluate the quality of estimate $\hat{\theta}$ using **mean squared error**,

$$\mathrm{MSE}(\hat{\theta}) = \mathbf{E}\left[(\hat{\theta} - \theta)^2\right] = \mathrm{bias}^2(\hat{\theta}) + \mathbf{Var}(\hat{\theta}) \quad , \mathrm{bias}(\hat{\theta}) = \mathrm{E}[\hat{\theta}] - \theta$$

- MSE for unbiased estimators is just,

$$\mathrm{MSE}(\hat{\theta}) = \mathbf{Var}(\hat{\theta})$$

- Bias-variance is fundamental tradeoff in statistical estimation
- MSE increases as **square** of bias
- Estimators with small bias (but low variance) can have lower MSE than unbiased estimators



Bias-Variance Decomposition

$$
\begin{aligned}
\mathrm{MSE}(\hat{\theta}) &= \mathbf{E}\left[(\hat{\theta}(X) - \theta)^2\right] \\
&= \mathbf{E}\left[\left(\hat{\theta} - \mathbf{E}[\hat{\theta}] + \mathbf{E}[\hat{\theta}] - \theta\right)^2\right] \\
&= \mathbf{E}[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2] + 2(\mathbf{E}[\hat{\theta}] - \theta)\mathbf{E}[\hat{\theta} - \mathbf{E}[\hat{\theta}]] + \mathbf{E}[(\mathbf{E}[\hat{\theta}] - \theta)^2] \\
&= \mathbf{E}[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2] + \left(\mathbf{E}[\hat{\theta}] - \theta\right)^2 \\
&= \mathrm{Var}(\hat{\theta}) + \mathrm{bias}^2(\hat{\theta})
\end{aligned}
$$

Other materials

- Bootstrap confidence intervals: https://math.mit.edu/~dav/05.dir/class24-prep-a.pdf

- Permutation test: https://www.jwilber.me/permutationtest/

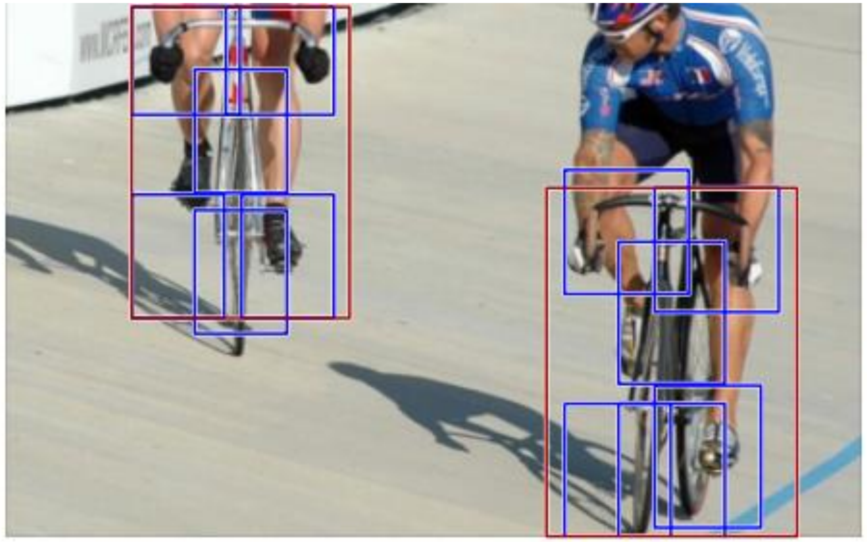
- STAT 566 lecture slides (at UA): https://www.math.arizona.edu/~jwatkins/stat566s20s.html

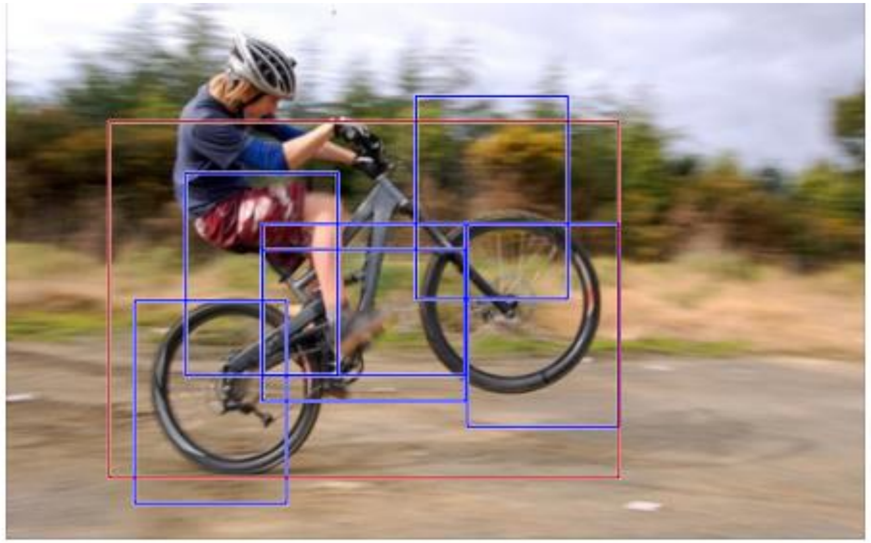
Next lecture (2/20)

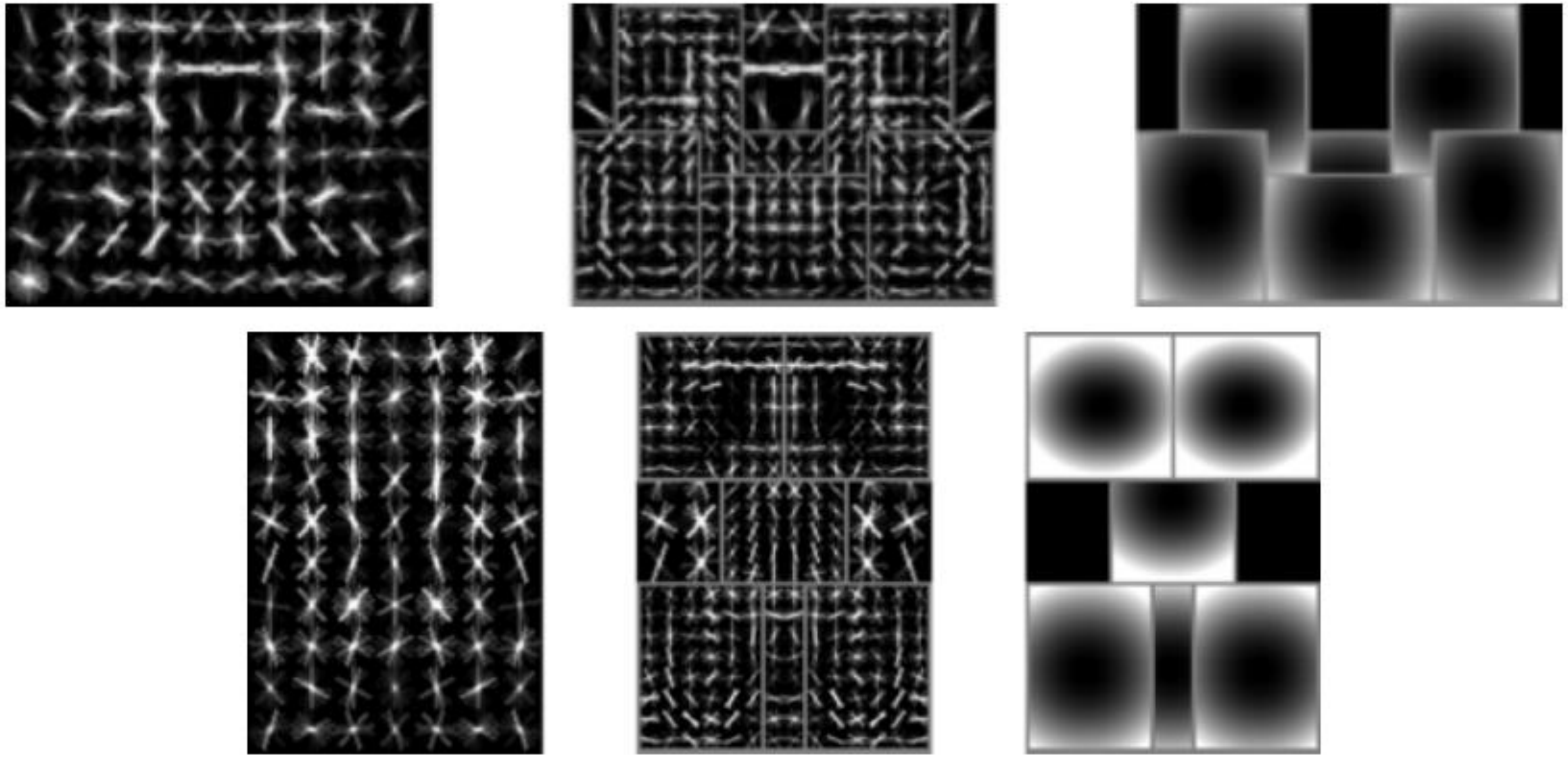
- Linear models revisited: classification, regression, loss minimization formulations

- Assigned reading: CIML Chapter 7

- *Note: We are skipping Chapter 6 for now!*







https://computervisionblog.wordpress.com/2016/01/23/distributed-code-or-grandmother-cells-insights-from-convolutional-neural-networks/

ROC curve examples

task: recognizing genomic units called operons

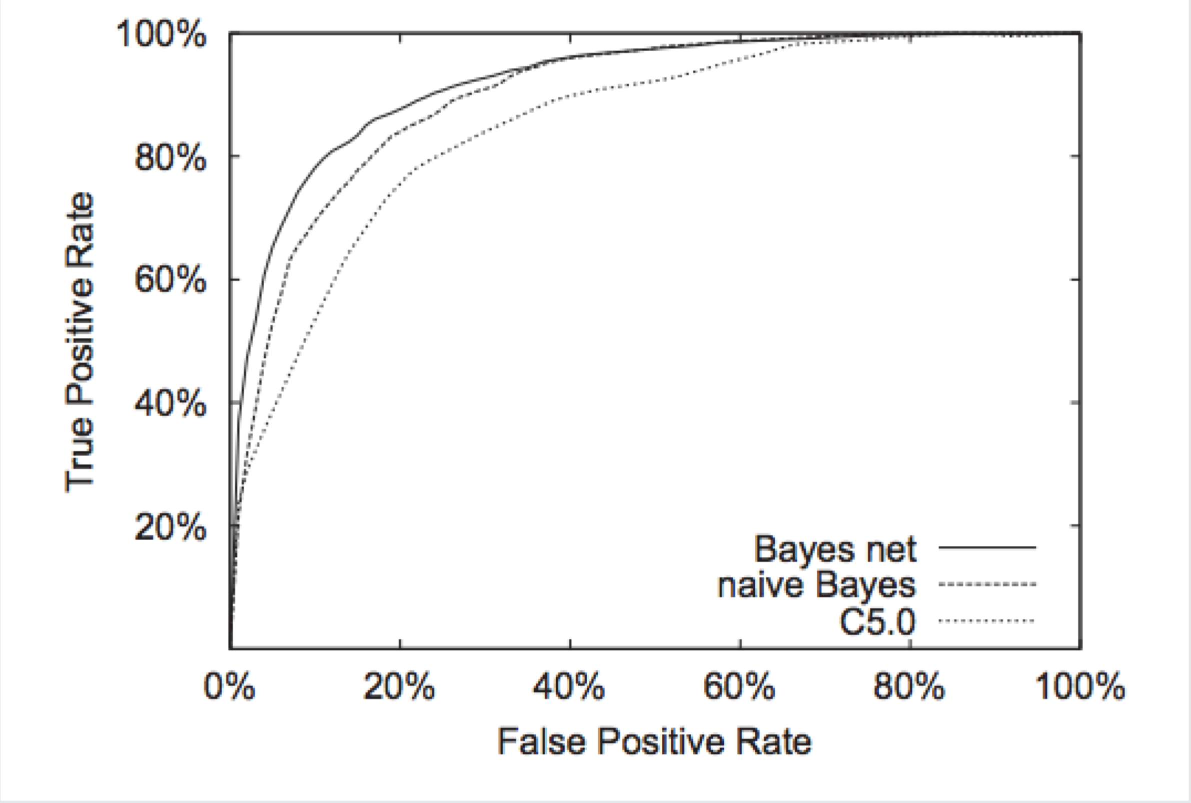
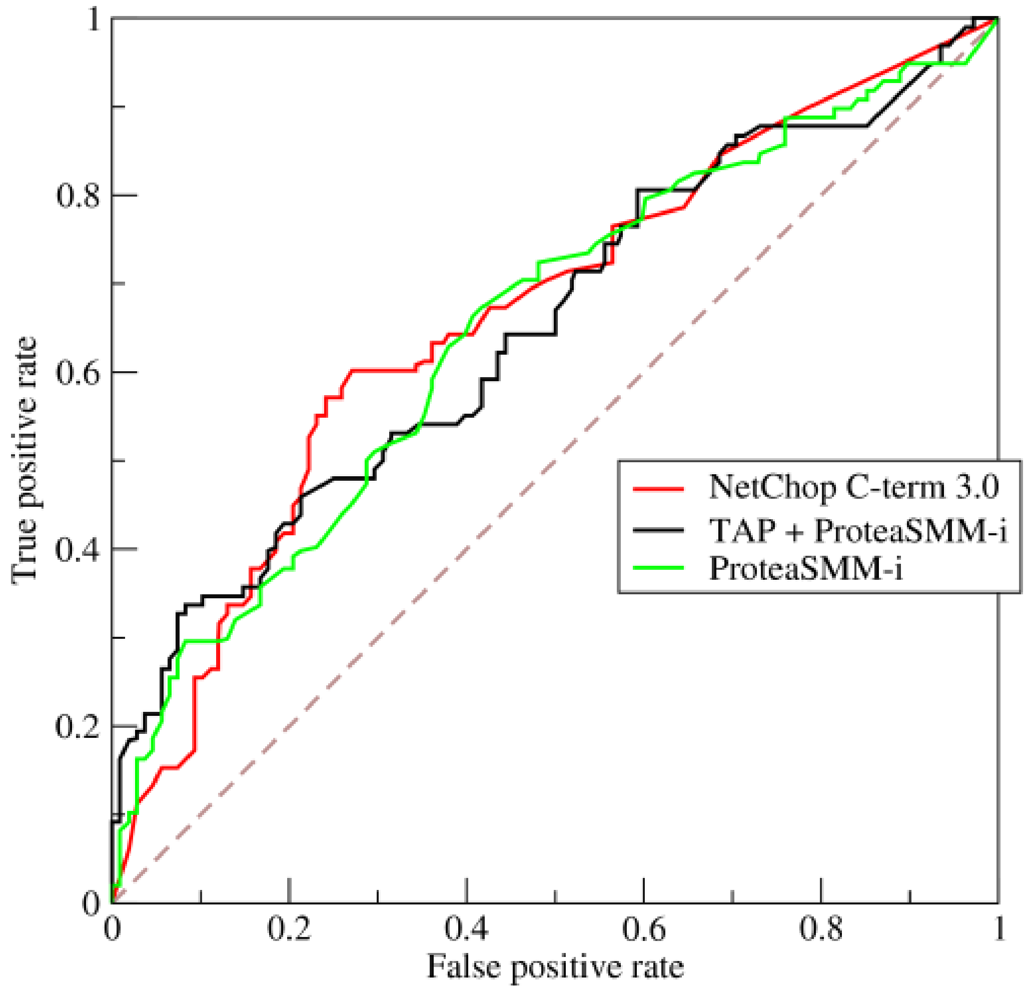


figure from Bockhorst et al., *Bioinformatics* 2003



from Wikipedia

Warning

Question How should we interpret a confidence interval (e.g. 95%)?

$$P(\theta \in (a(X), b(X))) \geq 0.95$$

Hint Think about what is random and what is not…

Warning

<u>Question</u> How should we interpret a confidence interval (e.g. 95%)?

$$P(\theta \in (a(X), b(X))) \geq 0.95$$

<u>Hint</u> Think about what is random and what is not…

This is NOT a probability statement about θ.

Warning

<u>Question</u> How should we interpret a confidence interval (e.g. 95%)?

$$P(\theta \in (a(X), b(X))) \geq 0.95$$

<u>Hint</u> Think about what is random and what is not…

Confidence interval: caveat in interpretation

- Suppose $\theta =$ average height of people in Tucson (in cm)

- Using a sample of size 100, we have computed a 95% confidence interval: [168, 171]

- Does that mean:

  with probability 0.95, $\theta \in$[168, 171] (*)

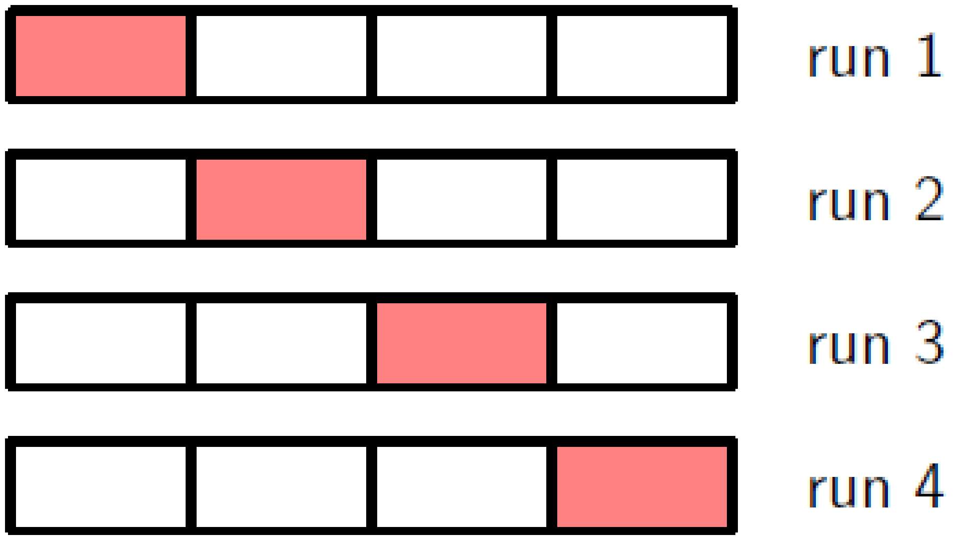
  ?

- **Hint**: What is the randomness in the above statement?

- **Answer:** The above interpretation is incorrect – "$\theta \in$[168, 171]" is either true or false – there is no randomness in this statement

- Then: what does $P(\theta \in (a(X), b(X))) \geq 0.95$ truly mean?

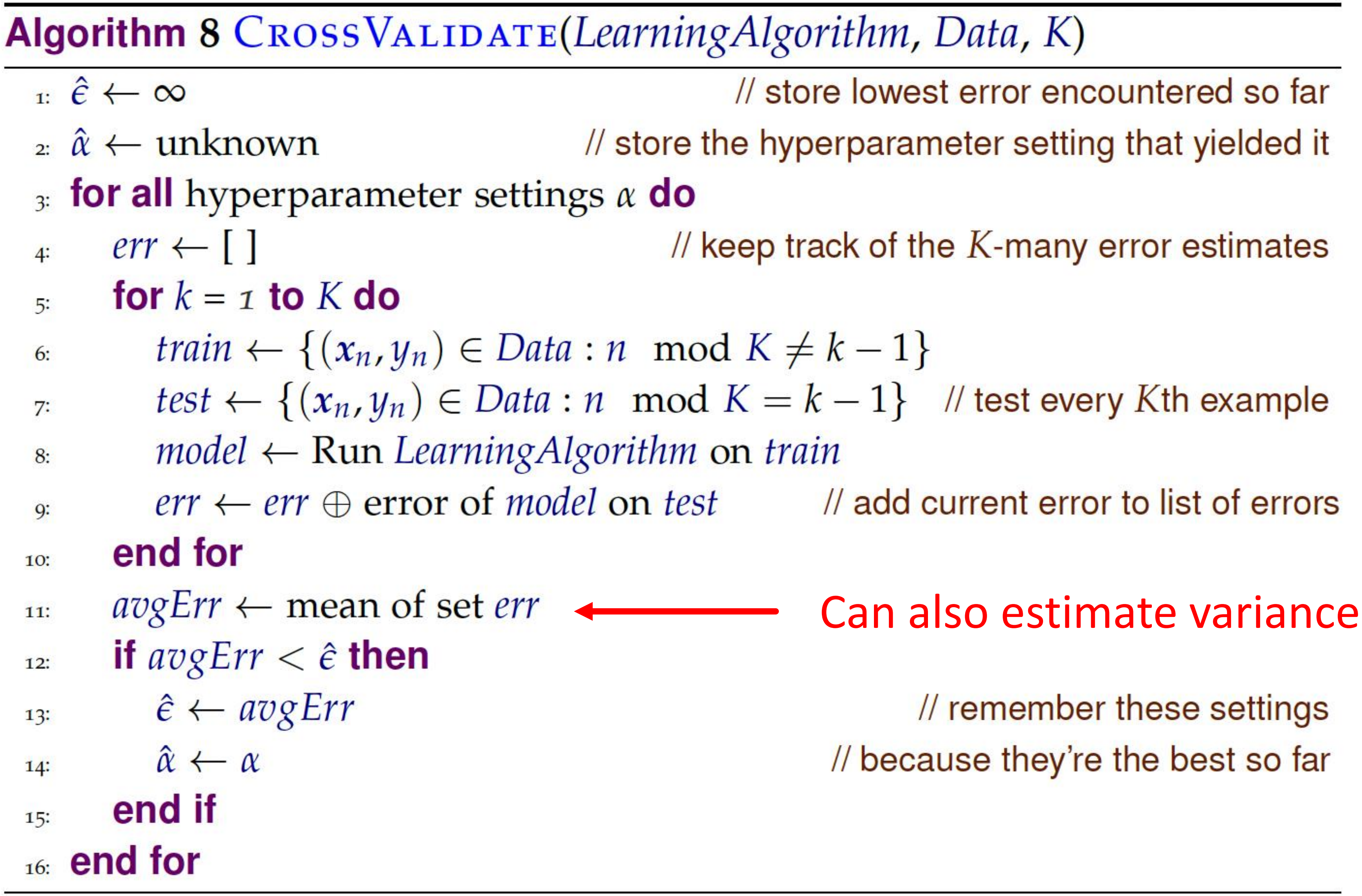
Cross-Validation



K-fold Cross Validation Partition training data into K “chunks” and for each run select one chunk to be validation data

For each run, fit to training data (K-1 chunks) and measure accuracy on validation set. Average model error across all runs. Estimate variance.

Source: Bishop, C. PRML



Algorithm 8 CrossValidate(*LearningAlgorithm*, *Data*, K)

```
$\hat{\epsilon} \leftarrow \infty$                                   // store lowest error encountered so far
$\hat{\alpha} \leftarrow$ unknown                          // store the hyperparameter setting that yielded it
for all hyperparameter settings $\alpha$ do
    $err \leftarrow [\,]$                                // keep track of the $K$-many error estimates
    for $k = 1$ to $K$ do
        $train \leftarrow \{(x_n, y_n) \in Data : n \mod K \neq k - 1\}$
        $test \leftarrow \{(x_n, y_n) \in Data : n \mod K = k - 1\}$   // test every $K$th example
        $model \leftarrow$ Run $LearningAlgorithm$ on $train$
        $err \leftarrow err \oplus$ error of $model$ on $test$       // add current error to list of errors
    end for
    $avgErr \leftarrow$ mean of set $err$
    if $avgErr < \hat{\epsilon}$ then
        $\hat{\epsilon} \leftarrow avgErr$                                // remember these settings
        $\hat{\alpha} \leftarrow \alpha$                                  // because they're the best so far
    end if
end for
```

Can also estimate variance here

Drawback Need to perform training K times for each model.

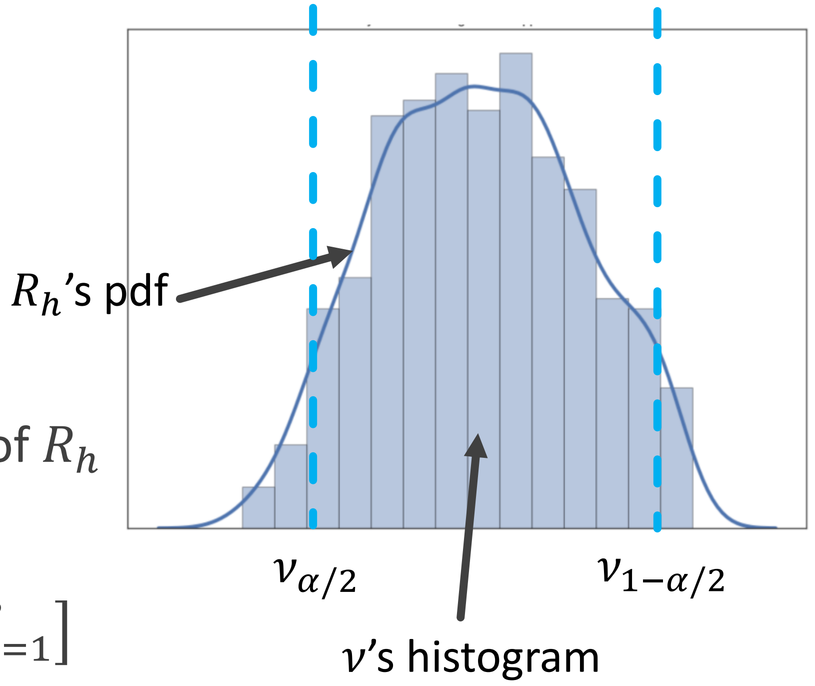
Bootstrapping CI

empirical distribution of $\{X_1, \ldots, X_n\}$: $\frac{1}{n}\sum_{i=1}^{n} \delta_{X_i}$ where δ_X is a dirac delta function

- Goal: estimate property h of D (:=$h(D)$) using confidence intervals, using sample S (e.g. h= F1 of model f)

- Idea: estimate the distribution of $h(S) - h(D)$, denoted by R_h

  by *bootstrapping* (resampling)
  - perform n times of "sampling with replacement" from S
  - repeat B times (e.g., B $= 100$) to obtain $S_1, \ldots, S_B$
  - take $\nu :=$ **empirical distribution** of $\{h(S_b) - h(S)\}_{b=1}^{B}$, as the **'shape'** of R_h



- Assumption: $h(S) - h(D) \sim R_h \approx \text{empirical\_distribution}\left[\{h(S_b) - h(S)\}_{b=1}^{B}\right]$

  Quantile interval: sort values and take top/bottom-quantiles (see next slide)

- With prob. $\approx 1 - \alpha$, $h(S) - h(D) \in [\nu_{\alpha/2}, \nu_{1-\alpha/2}] \Longrightarrow I(S) = [h(S) - \nu_{1-\alpha/2}, h(S) - \nu_{\alpha/2}]$